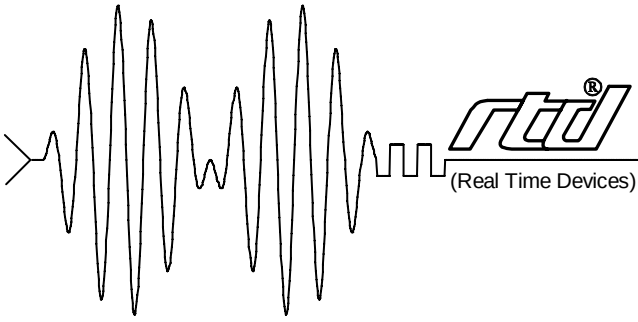
DM210/DM5210
User's Manual



RTD Embedded Technologies Inc.

"Accessing the Analog World"®

AS9100 and ISO 9001 Certified

BDM-610010022
Rev. B

DM210/DM5210
User's Manual

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

Rev. A New manual naming method
Rev. B Changed pictures to match Rev B PCB.

Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

LIST OF ILLUSTRATIONS **3**

INTRODUCTION i-1

Analog-to-Digital Conversion i-3
8254 Timer/Counter i-3
Digital I/O i-3
What Comes With Your Module i-3
Module Accessories i-4
Application Software and Drivers i-4
Hardware Accessories i-4
Optional Configurations i-4
Using This Manual i-4
When You Need Help i-4

CHAPTER 1 MODULE SETTINGS 1-1

Factory-Configured Switch and Jumper Settings 3
P3 — Interrupt Channel Select (Factory Setting: Interrupt Channels Disabled; G Connected) 4
P4 — 8254 Timer/Counter Clock Sources (Factory Settings: CLK0-OSC, CLK1-OT0, CLK2-OT1) 5
P5 — Analog Input Voltage Range (Factory Setting: 10V) 6
P6 — Analog Input Voltage Polarity (Factory Setting: +/- (Bipolar)) 6
P7 — Interrupt Source (Factory Setting: OT2) 6
P8 — 8255 Port B, Bits 4-7 Pads (Factory Setting: No Connections) 6
S1 — Base Address (Factory Setting: 300 hex (768 decimal)) 7
Pull-up/Pull-down Resistors on Digital I/O Lines 8
Gx, Resistor Configurable Gain 9

CHAPTER 2 MODULE INSTALLATION 2-1

Module Installation 2-3
External I/O Connections 2-3
Connecting the Analog Inputs 2-4
Connecting the Timer/Counters and Digital I/O 2-4
Running the 5210DIAG Diagnostics Program 2-4

CHAPTER 3 HARDWARE DESCRIPTION 3-1

A/D Conversion Circuitry 3-3
Analog Inputs 3-3
A/D Converter 3-3
Timer/Counters 3-4
Digital I/O, Programmable Peripheral Interface 3-4
Interrupts 3-4

CHAPTER 4 MODULE OPERATION AND PROGRAMMING 4-1

Defining the I/O Map 4-3
BA + 0: PPI Port A — Digital I/O (Read/Write) 4-3
BA + 1: PPI Port B — Channel Select (Read/Write) 4-3

BA + 2: PPI Port C — Digital I/O (Read/Write) 4-4
BA + 3: 8255 PPI Control Word (Write Only) 4-4
BA + 4: 8254 Timer/Counter 0 (Read/Write) 4-6
BA + 5: 8254 Timer/Counter 1 (Read/Write) 4-6
BA + 6: 8254 Timer/Counter 2 (Read/Write) 4-6
BA + 7: 8254 Control Word (Write Only) 4-6
BA + 8: Read MSB Data/Start 12-Bit Conversion (Read/Write) 4-7
BA + 9: Read LSB Data/Start 8-Bit Conversion (Read/Write) 4-7
BA + 10: Read Status/Clear IRQ (Read/Write) 4-7
BA + 11: IRQ Enable (Write Only) 4-7
Programming the DM210/DM5210 4-8
Clearing and Setting Bits in a Port 4-8
A/D Conversions 4-10
• Initializing the 8255 PPI 4-10
• Selecting a Channel 4-10
• Enabling and Disabling Interrupts 4-10
• Starting an A/D Conversion 4-10
• Channel Scanning 4-10
• Monitoring Conversion Status 4-11
• Reading the Converted Data 4-11
Interrupts 4-12
• What Is an Interrupt? 4-12
• Interrupt Request Lines 4-12
• 8259 Programmable Interrupt Controller 4-13
• Interrupt Mask Register (IMR) 4-13
• End-of-Interrupt (EOI) Command 4-13
• What Exactly Happens When an Interrupt Occurs? 4-13
• Using Interrupts in Your Programs 4-13
• Writing an Interrupt Service Routine (ISR) 4-13
• Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector 4-15
• Restoring the Startup IMR and Interrupt Vector 4-15
• Common Interrupt Mistakes 4-15
Timer/Counters 4-16
Digital I/O 4-17
Example Programs and Flow Diagrams 4-18
C Programs 4-18
BASIC Programs 4-18
Flow Diagrams 4-18
• Single Convert Flow Diagram (Figure 4-3) 4-19
• Channel Scanning Flow Diagram (Figure 4-4) 4-19

CHAPTER 5 CALIBRATION 5-1

Required Equipment 5-3
A/D Calibration 5-4
Unipolar Calibration 5-4
Bipolar Calibration 5-5

APPENDIX A DM210/DM5210 SPECIFICATIONS A-1

APPENDIX B P2 CONNECTOR PIN ASSIGNMENTS B-1

APPENDIX C COMPONENT DATA SHEETS C-1

APPENDIX D WARRANTY AND RETURN POLICY D-1

List of Illustrations

1-1 Module Layout Showing Factory-Configured Settings 1-3
1-2 Interrupt Channel Jumper, P3 1-4
1-3 Pulling Down the Interrupt Request Line 1-4
1-4 8254 Timer/Counter Clock Source Jumpers, P4 1-5
1-5 8254 Timer/Counter Circuit Block Diagram 1-5
1-6 Analog Input Voltage Range and Polarity, P5 and P6 1-6
1-7 Interrupt Source Jumper, P7 1-6
1-8 Port B, Bits 4-7 Pads, P8 1-6
1-9 Base Address Switch, S1 1-7
1-10 Pull-up/Pull-down Resistors for the 8255 1-8
1-11 Adding Pull-ups and Pull-downs to Digital I/O Lines 1-9
1-12 Gain Circuitry and Formulas for Calculating Gx and f 1-10
1-13 Diagram for Removal of Solder Short 1-10
2-1 P2 I/O Connector Pin Assignments 2-4
2-2 Analog Input Connections 2-5
3-1 DM210/DM5210 Block Diagram 3-3
4-1 A/D Conversion Timing Diagram 4-11
4-2 8254 Programmable Interval Timer Circuit Block Diagram 4-16
4-3 Single Conversion Flow Diagram 4-19
4-4 Channel Scanning Flow Diagram 4-20
5-1 Module Layout 5-3

INTRODUCTION

The DM210 and DM5210 dataModule® medium speed analog input modules turn your IBM PC-compatible cpuModule™ or other PC/104 computer into a high-performance data acquisition and control system. Ultra-compact for embedded and portable applications, the 210/5210 features:

- 16 single-ended analog input channels,
- 12-bit, 20 microsecond A/D converter,
- ±5, ±10, or 0 to +10 volt analog input range,
- Resistor configurable gain,
- 20 TTL/CMOS 8255 based programmable digital I/O (16 at I/O connector, 4 at on-board pads),
- Three independent 16-bit, 8-MHz timer/counters,
- +5 volt only operation (DM5210),
- BASIC and Turbo C source code; diagnostics program.

Note that the difference between the DM210 and DM5210 is the power supply requirements: the DM210 requires ±12 and +5 volt power sources and the DM5210 requires a +5 volt source. The following paragraphs briefly describe the major functions of the module. A detailed discussion of module functions is included in subsequent chapters.

Analog-to-Digital Conversion

The analog-to-digital (A/D) circuitry receives up to 16 single-ended analog inputs and converts these inputs into 12-bit digital data words which can then be read and/or transferred to PC memory.

The analog input voltage range is jumper-selectable for bipolar ranges of -5 to +5 volts or -10 to +10 volts, or a unipolar range of 0 to +10 volts. The module is factory set for -5 to +5 volts. Overvoltage protection to ±35 volts is provided at the inputs. A/D conversions are performed by a 12-bit successive approximation converter. This high-performance converter and the high-speed sample-and-hold amplifier preceding it make sure that dynamic input voltages are accurately digitized. The resolution of a 12-bit conversion is 2.4414 millivolts and the maximum throughput is 40,000 samples per second.

The converted data is read and/or transferred to PC memory, one byte at a time, through the PC data bus.

8254 Timer/Counter

An 8254 programmable interval timer contains three 16-bit, 8-MHz timer/counters to support a wide range of timing and counting functions. The clock, gate and output pins for each of the three timer/counters are available at the I/O connector.

Digital I/O

The 210/5210 has 20 TTL/CMOS-compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. The lines are provided by the on-board 8255 programmable peripheral interface (PPI) chip. Sixteen of the lines are brought out to the I/O connector and four are available at a set of on-board pads located near the edge of the module for easy access. Pads for installing and activating pull-up or pull-down resistors are included on the module for the 16 lines brought out to the I/O connector. Installation procedures are given at the end of Chapter 1, *Module Settings.*

What Comes With Your Module

You receive the following items in your 210/5210 package:

- DM210 or DM5210 interface module with stackthrough bus header
- Mounting hardware
- Software and diagnostics diskette with example programs in BASIC and Turbo C; source code
- User's manual

If any item is missing or damaged, please call RTD Embedded Technologies, Inc. Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

Module Accessories

In addition to the items included in your module package, RTD Embedded Technologies, Inc, offers a full line of software and hardware accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your module's application.

Application Software and Drivers

Our custom application software packages provide excellent data acquisition and analysis support. Use SIGNAL*VIEW™ for real-time monitoring and data acquisition, and SIGNAL*MATH™ for integrated data acquisition and sophisticated digital signal processing and analysis. rtdLinx™ drivers provide full-featured high level interfaces between the 210/5210 and custom or third party software. rtdLinx source code is available for a one-time nominal fee.

Hardware Accessories

Hardware accessories for the 210/5210 include the TMX32 analog input expansion board with thermocouple compensation which can expand a single input channel on your 210/5210 to 16 differential or 32 single-ended input channels, the OP series optoisolated digital input boards, the MR series mechanical relay output boards, the OR16 optoisolated digital input/mechanical relay output board, the USF8 universal sensor interface with sensor excitation, the TS16 thermocouple sensor board, the TB50 terminal board and XB50 prototype/terminal board for easy signal access and prototype development, the DM14 extender board for testing your module in a conventional desktop computer, and XP50 flat ribbon cable assembly for external interfacing.

Optional Configurations

Other configurations of the 210/5210 are available, such as vertical connectors on some or all I/O connectors, a right angle or other type of connector for easy use of the four digital I/O lines brought out to pads, or a non-stackthrough bus connector. If you need an optional configuration for your requirements, please consult the factory.

Using This Manual

This manual is intended to help you install your new module and get it running quickly, while also providing enough detail about the module and its functions so that you can enjoy maximum use of its features even in the most complex applications. We assume that you already have an understanding of data acquisition principles and that you can customize the example software or write your own applications programs.

When You Need Help

This manual and the example programs in the software package included with your module provide enough information to properly use all of the module's features. If you have any problems installing or using this module, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1

MODULE SETTINGS

The 210/5210 has jumper and switch settings you can change if necessary for your application. The module is factory-configured with the settings listed in Table 1-1 and shown on the module diagram at the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the module in your system.

By installing resistor packs and soldering jumpers in the desired locations in the associated pads as described near the end of the chapter, you can configure 16 of your digital I/O lines to be pulled up or pulled down.

The final section describes how to install two resistors and a trimpot to set the resistor configurable gain to the value required for your application. A pad for installing a capacitor is also included in the gain circuitry for creating a low-pass filter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switch on the 210/5210. Figure 1-1 shows the module layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your module in your system.

Table 1-1: Factory Settings

Switch/ Jumper	Function Controlled	Factory Settings (Jumpers Installed)
P3	Selects the active interrupt channel; pulls tri-state buffer to ground (G) for multiple interrupt applications	Interrupt channels disabled; jumper installed on G (ground for buffer)
P4	Sets the clock sources for the 8254 timer/counters (TC0-TC2)	Jumpers installed on CLK0-OSC, CLK1-OT0 & CLK2-OT1 (cascaded)
P5	Sets the analog input voltage range	10V
P6	Sets the analog input voltage polarity	+/-
P7	Selects one of three signals as the interrupt source	OT2
P8	8255 Port B, bits 4-7, pads for user connections	No connections installed
S1	Sets the base address	300 hex (768 decimal)

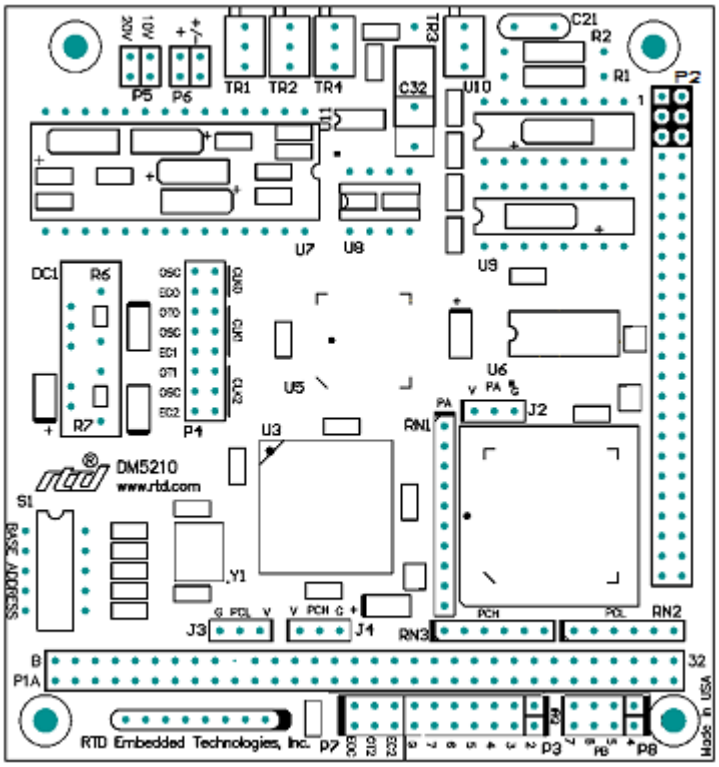


Fig. 1-1 — Module Layout Showing Factory-Configured Settings

P3 — Interrupt Channel Select (Factory Setting: Interrupt Channels Disabled; G Connected)

This header connector, shown in Figure 1-2, lets you connect any one of the three interrupt sources on P7 to an interrupt channel, IRQ2 (highest priority channel) through IRQ7 (lowest priority channel). IRQ2 is the righttmost channel and IRQ7 is the leftmost channel (next to last pair of pins). To activate a channel, you must install a jumper vertically across the desired IRQ channel's pair of pins. Figure 1-2a shows the factory setting; Figure 1-2b shows the interrupt source connected to IRQ3.

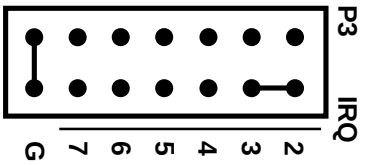


Fig. 1-2a:
Factory Setting

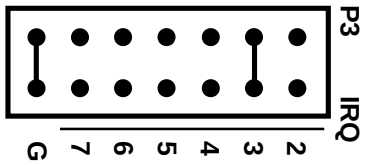


Fig. 1-2b: Interrupt Source
Connected to IRQ3

Fig. 1-2 — Interrupt Channel Jumper, P3

This module supports an interrupt sharing mode where the pins labeled G, connect a 1 kilohm pull-down resistor to the output of a high-impedance tri-state driver which carries the interrupt request signal. This pull-down resistor pulls the interrupt request line low whenever interrupts are not active. Whenever an interrupt request is made, the tri-state buffer is enabled, forcing the output high and generating an interrupt. You can monitor the interrupt status through bit 1 in the status word (I/O address location BA + 10). After the interrupt has been serviced, the reset command returns the IRQ line low, disabling the tri-state buffer, and pulling the output low again. Figure 1-3 shows this circuit. Because the interrupt request line is pulled low only by the pull-down resistor, you can have two or more modules which share the same IRQ channel. You can tell which module issued the interrupt request by monitoring each module's IRQ status bit. If you are not planning on sharing interrupts or if you are not sure that your CPU supports interrupt sharing, it is best to disable this feature and use the interrupts in the normal mode. This will insure compatibility with all CPUs. See chapter 4 for details on disabling the interrupt sharing circuit.

NOTE: When you use multiple modules that share the same interrupt, only one module should have the G jumper installed. The rest should be disconnected. Whenever you operate a single module, the G jumper should be installed. Whenever you operate the module with interrupt sharing disabled, the G jumper should be removed.

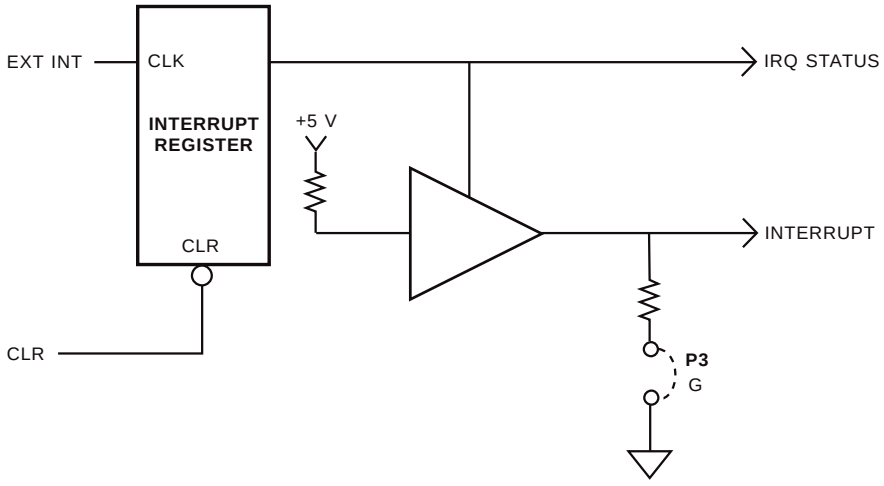


Fig. 1-3 — Pulling Down the Interrupt Request Line

P4 — 8254 Timer/Counter Clock Sources (Factory Settings: CLK0-OSC, CLK1-OT0, CLK2-OT1)

This header connector, shown in Figure 1-4, lets you select the clock sources for the 8254 timer/counters, TC0, TC1, and TC2. The factory setting cascades all three timer/counters, with the clock source for TC0 being the on-board 8 MHz oscillator, the output of TC0 providing the clock for TC1, and the output of TC1 providing the clock for TC2. You can connect any or all of the sources to an external clock input through the P2 I/O connector, or you can set TC1 and TC2 to be clocked by the 8 MHz oscillator. Figure 1-5 shows a block diagram of the timer/counter circuitry to help you with these connections.

NOTE: When installing jumpers on this header, make sure that only one jumper is installed in each group of two or three CLK pins.

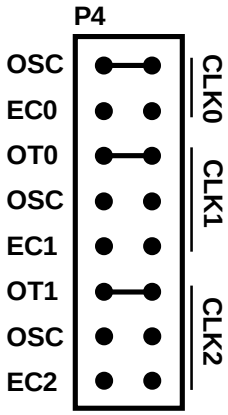


Fig. 1-4 — 8254 Timer/Counter Clock Source Jumpers, P4

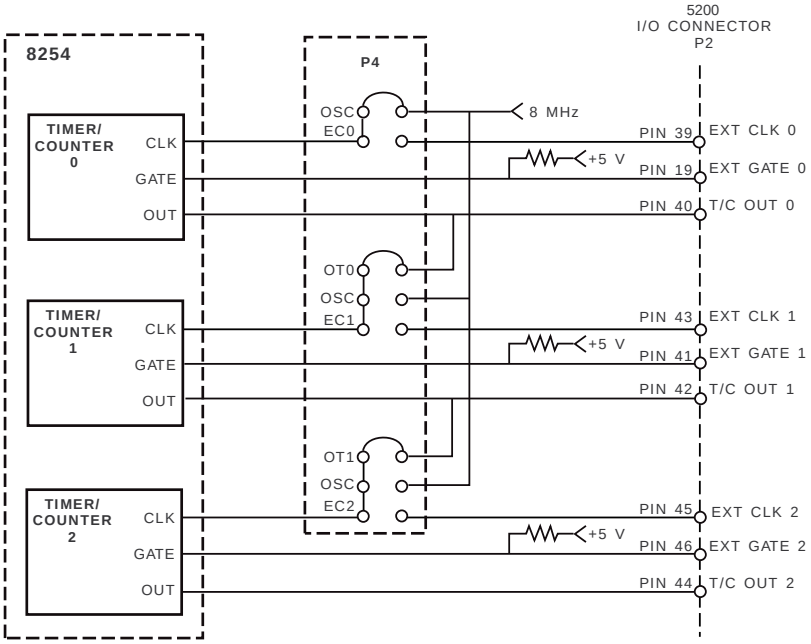


Fig. 1-5 — 8254 Timer/Counter Circuit Block Diagram

P5 — Analog Input Voltage Range (Factory Setting: 10V)

This header connector, shown in Figure 1-6, lets you select the analog input voltage range. The range is set by placing the jumper across the pair of pins labeled 10V, giving you a 10 volt range, or by placing the jumper across the pins labeled 20V, giving you a 20-volt range. Note that when you place a jumper across 20V, you must place the jumper on P6 across the +/- pins (bipolar range of -10 to +10 volts). The + setting on P6 cannot be used with 20V.

P6 — Analog Input Voltage Polarity (Factory Setting: +/- (Bipolar))

This header connector, shown in Figure 1-6, lets you select the analog input polarity by placing a jumper across the pins labeled + for 0 to +10 volts, or +/- for ±5 or ±10 volts. Note that when you place a jumper across 20V on P5, you must place the P6 jumper across +/- (±10 volts). The + setting cannot be used with the 20 volt input range. Figure 1-6 shows the three possible input voltage configurations for P5 and P6.

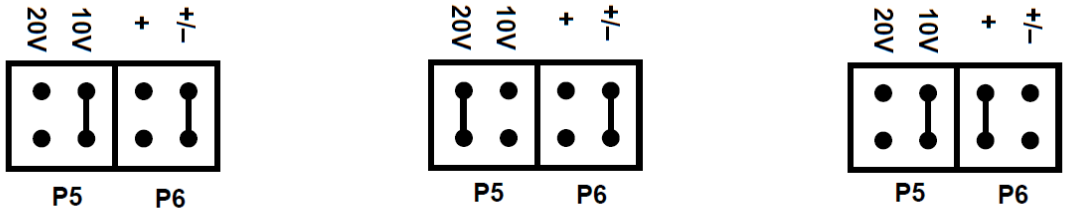


Fig. 1-6a: Factory Setting, ±5V

Fig. 1-6b: Inputs Connected for ±10V

Fig. 1-6c: Inputs Connected for 0 to +10V

Fig. 1-6 — Analog Input Voltage Range and Polarity, P5 and P6

P7 — Interrupt Source (Factory Setting: OT2)

This header connector, shown in Figure 1-7, lets you select any one of three signal sources for use in generating an interrupt. An interrupt source is chosen by placing a jumper across the desired pair of pins. The interrupt sources available are the A/D end-of-convert (EOC), the output of timer/counter 2 (OT2), and timer/counter external clock 2 (EC2). The interrupt channel for the selected source is set on P3.

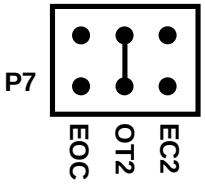


Fig. 1-7— Interrupt Source Jumper, P7

P8 — 8255 Port B, Bits 4-7 Pads (Factory Setting: No Connections)

These four pads, shown in Figure 1-8, provide easy access to the top four bits of Port Bin the 8255 PPI. These bits are available to the user as digital outputs. You can install a header, right angle connector, or use another method to connect these signals into your circuit. The holes closest to the edge of the board are the ground, and the holes closest to the bus connector are the signal side. The bottom four bits of Port B are reserved for on-board functions.

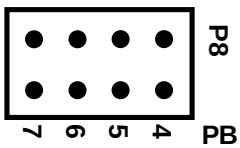


Fig. 1-8— Port B, Bits 4-7 Pads, P8

S1 — Base Address (Factory Setting: 300 hex (768 decimal))

One of the most common causes of failure when you are first trying your module is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the 210/5210 attempts to use I/O address locations already used by another device, contention results and the module does not work.

To avoid this problem, the 210/5210 has an easily accessible DIP switch, S1, which lets you select any one of 32 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any one of the values listed in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 5) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your module, record the value in the table inside the back cover. Figure 1-9 shows the DIP switch set for a base address of 300 hex (768 decimal).

Table 1-2: Base Address Switch Settings, S1

Base Address Decimal / (Hex)	Switch Setting 5 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 5 4 3 2 1
512 / (200)	0 0 0 0 0	768 / (300)	1 0 0 0 0
528 / (210)	0 0 0 0 1	784 / (310)	1 0 0 0 1
544 / (220)	0 0 0 1 0	800 / (320)	1 0 0 1 0
560 / (230)	0 0 0 1 1	816 / (330)	1 0 0 1 1
576 / (240)	0 0 1 0 0	832 / (340)	1 0 1 0 0
592 / (250)	0 0 1 0 1	848 / (350)	1 0 1 0 1
608 / (260)	0 0 1 1 0	864 / (360)	1 0 1 1 0
624 / (270)	0 0 1 1 1	880 / (370)	1 0 1 1 1
640 / (280)	0 1 0 0 0	896 / (380)	1 1 0 0 0
656 / (290)	0 1 0 0 1	912 / (390)	1 1 0 0 1
672 / (2A0)	0 1 0 1 0	928 / (3A0)	1 1 0 1 0
688 / (2B0)	0 1 0 1 1	944 / (3B0)	1 1 0 1 1
704 / (2C0)	0 1 1 0 0	960 / (3C0)	1 1 1 0 0
720 / (2D0)	0 1 1 0 1	976 / (3D0)	1 1 1 0 1
736 / (2E0)	0 1 1 1 0	992 / (3E0)	1 1 1 1 0
752 / (2F0)	0 1 1 1 1	1008 / (3F0)	1 1 1 1 1
0 = closed, 1 = open			

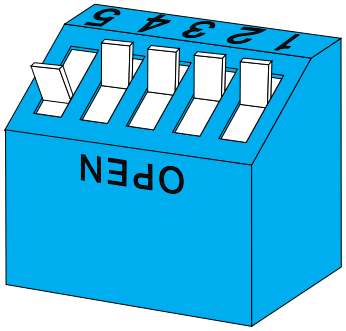


Fig. 1-9 — Base Address Switch, S1

Pull-up/Pull-down Resistors on Digital I/O Lines

The 8255 programmable peripheral interface provides 20 TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. These lines are divided into four groups: eight Port A lines, four upper Port B lines (the four lower lines are used to control board functions), four Port C Lower lines, and four Port C Upper lines. The 16 lines of Ports A and C are available at the P2 I/O connector. You can install and connect pull-up or pull-down resistors for these 16 lines as described below. For example, you may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull lines down for connection to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high. The Port A lines of the 8255 automatically power up as inputs - which can float high - during the few moments before the board is first initialized. This can cause the external devices connected to these lines to operate erratically. By pulling these lines down, when the data acquisition system is first turned on, the motors will not switch on before the 8255 is initialized.

To use the pull-up/pull-down feature, you must first install single in-line resistor packs in any or all of the three locations around the 8255, labeled PA (Port A), PCL (Port C lower), and PCH (Port C upper). The four Port B lines cannot be pulled up or down by installing resistor packs. PA takes a 10-pin pack, and CL and CH take 6-pin packs. Figure 1-10 shows this circuitry.

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. Locate the three-hole pads on the module near the resistor packs. They are labeled G (for ground) on one end and V (for +5V) on the other end. The middle hole is common. PA is for Port A, CL is for Port C Lower, and CH is for Port C Upper. To operate as pull-ups, solder a jumper wire between the common pin (middle pin of the three) and the V pin. For pull-downs, solder a jumper wire between the common pin (middle pin) and the G pin. Figure 1-11 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

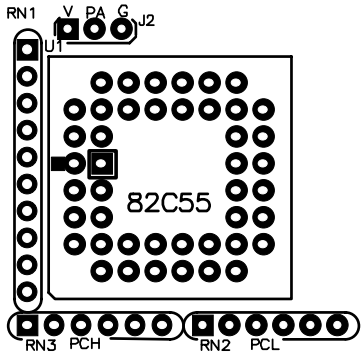


Fig. 1-10—Pull-up/Pull-down Resistors for the 8255

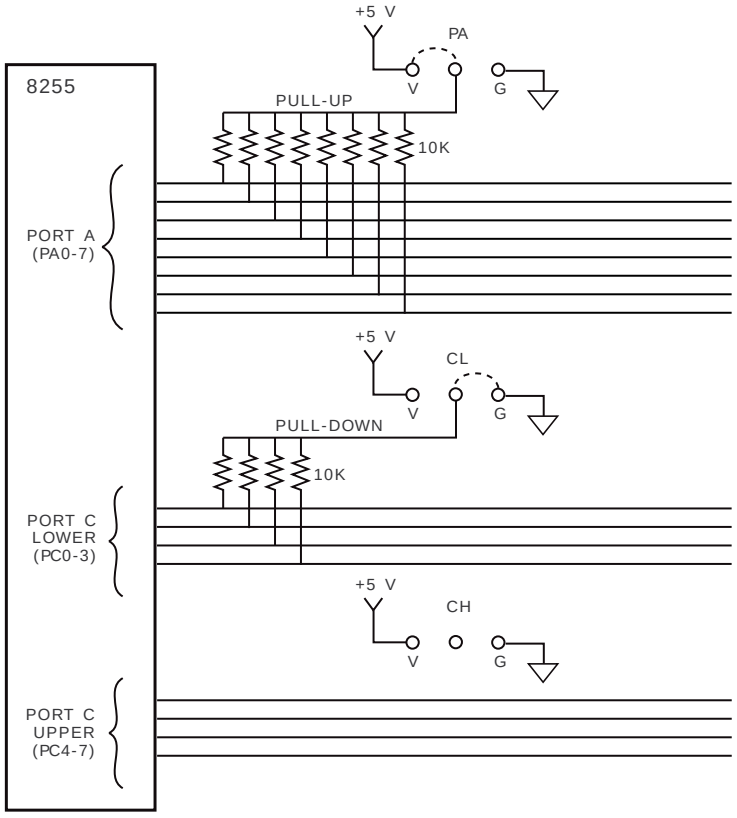


Fig. 1-11 — Adding Pull-ups and Pull-downs to Digital I/O Lines

Gx, Resistor Configurable Gain

The 210/5210 has a resistor configurable gain circuitry, Gx, so that you can easily configure special gain settings for a specific application. Note that when you use this feature, all of the input channels will operate only at your custom gain setting. Gx is derived by adding resistors R1 and R2, trimpot TR3, and capacitor C21, all located in the upper right area of the module. The resistors and trimpot combine to set the gain, as shown in the formula in Figure 1-12. Capacitor C21 is provided so that you can add low-pass filtering in the gain circuit. If your input signal is a slowly changing one and you do not need to measure it at a higher rate, you may want to add a capacitor at C21 in order to reduce the input frequency range and in turn reduce the noise on your input signal. The formula for setting the frequency is given in the diagram. Figure 1-12 shows how the Gx circuitry is configured.

As shown in Figure 1-12, a solder short must be removed from the module to activate the Gx circuitry. This short is located on the **bottom side** of the module under U11 (AD712 IC). Figure 1-13 shows the location of the solder short.

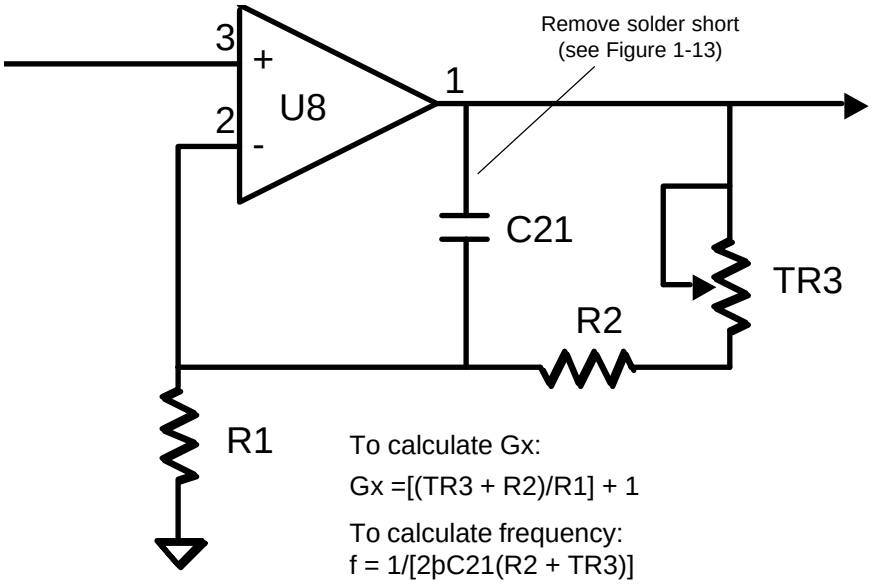


Fig. 1-12 — Gain Circuitry and Formulas for Calculating Gx and f

Remove solder short JS2 from bottom side of module

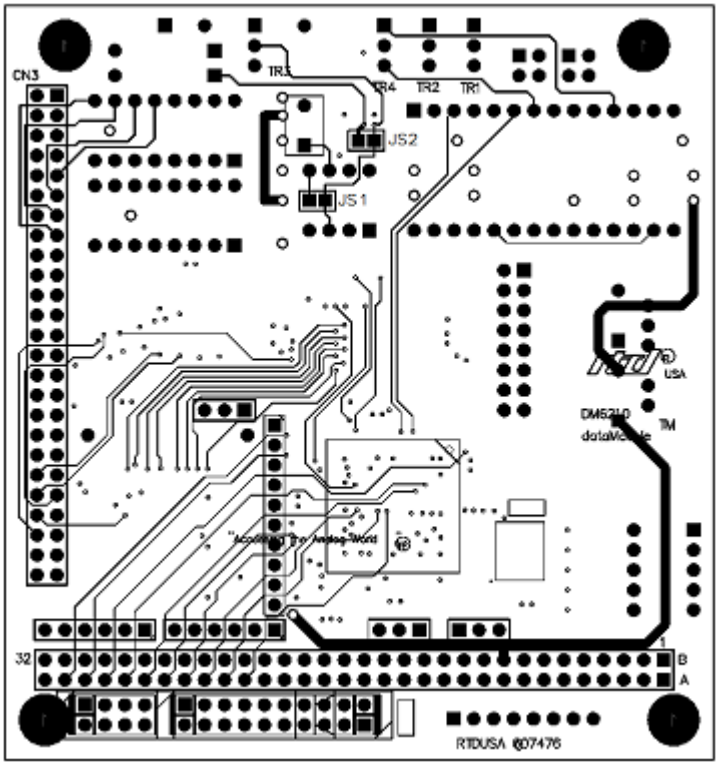


Fig. 1-13 — Diagram for Removal of Solder Short

CHAPTER 2

MODULE INSTALLATION

The 210/5210 is easy to install in your cpuModule™ or other PC/104 based system. This chapter tells you step-by-step how to install and connect the module.

After you have installed the module and made all of your connections, you can turn your system on and run the 5210DIAG diagnostics program included on your example software disk to verify that your module is working.

Module Installation

Keep the module in its antistatic bag until you are ready to install it in your cpuModule™ or other PC/104 based system. When removing it from the bag, hold the module at the edges and do not touch the components or connectors.

Before installing the module in your system, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable module operation and erratic response.

The 210/5210 comes with a stackthrough P1 connector. The stackthrough connector lets you stack another module on top of your 210/5210.

To install the module, follow the procedures described in the computer manual and the steps below:

1. Turn OFF the power to your system.
2. Touch a metal rack to discharge any static buildup and then remove the module from its antistatic bag.
3. Select the appropriate standoffs for your application to secure the module when you install it in your system (two sizes are included with the module).
4. Holding the module by its edges, orient it so that the P1 bus connector's pin 1 lines up with pin 1 of the expansion connector onto which you are installing the module.
5. After carefully positioning the module so that the pins are lined up and resting on the expansion connector, gently and evenly press down on the module until it is secured on the connector.

   NOTE: Do not force the module onto the connector. If the module does not readily press into place, remove it and try again. Wiggling the module or exerting too much pressure can result in damage to the 210/5210 or to the mating module.
6. After the module is installed, connect the cable to I/O connector P2 on the module. When making this connection, note that there is no keying to guide you in orientation. You must make sure that pin 1 of the cable is connected to pin 1 of P2 (pin 1 is marked on the module with a small square). For twisted pair cables, pin 1 is the dark brown wire; for standard single wire cables, pin 1 is the red wire.
7. Make sure all connections are secure.

External I/O Connections

Figure 2-1 shows the 210/5210's P2 I/O connector pinout. Refer to this diagram as you make your I/O connections. Note that the +12 and -12 volt signals are available at pins 47 and 49 only if your computer supplies these voltages.

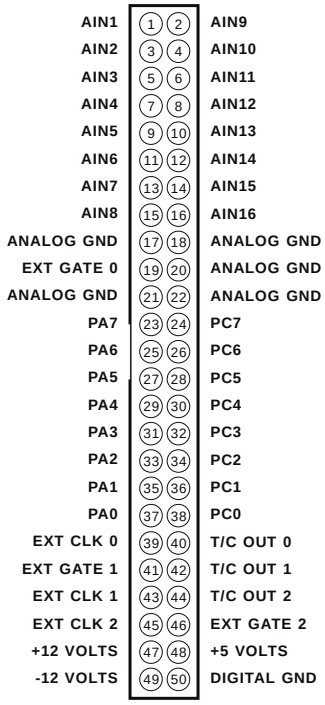

AIN1 1 2 AIN9
AIN2 3 4 AIN10
AIN3 5 6 AIN11
AIN4 7 8 AIN12
AIN5 9 10 AIN13
AIN6 11 12 AIN14
AIN7 13 14 AIN15
AIN8 15 16 AIN16
ANALOG GND 17 18 ANALOG GND
EXT GATE 0 19 20 ANALOG GND
ANALOG GND 21 22 ANALOG GND
PA7 23 24 PC7
PA6 25 26 PC6
PA5 27 28 PC5
PA4 29 30 PC4
PA3 31 32 PC3
PA2 33 34 PC2
PA1 35 36 PC1
PA0 37 38 PC0
EXT CLK 0 39 40 T/C OUT 0
EXT GATE 1 41 42 T/C OUT 1
EXT CLK 1 43 44 T/C OUT 2
EXT CLK 2 45 46 EXT GATE 2
+12 VOLTS 47 48 +5 VOLTS
-12 VOLTS 49 50 DIGITAL GND

Fig. 2-1 — P2 I/O Connector Pin Assignments

Connecting the Analog Inputs

NOTE: It is good practice to connect all unused channels to ground, as shown in the following diagram. Failure to do so may affect the accuracy of your results.

Connect the high side of the analog input to one of the analog input channels, AIN1 through AIN16, and connect the low side to the corresponding dedicated ANALOG GND for the selected channel. Figure 2-2 shows how these connections are made.

Connecting the Timer/Counters and Digital I/O

For all of these connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to any DIGITAL GND.

Running the 5210DIAG Diagnostics Program

Now that your module is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 5210DIAG, is included with your example software to help you verify your module's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

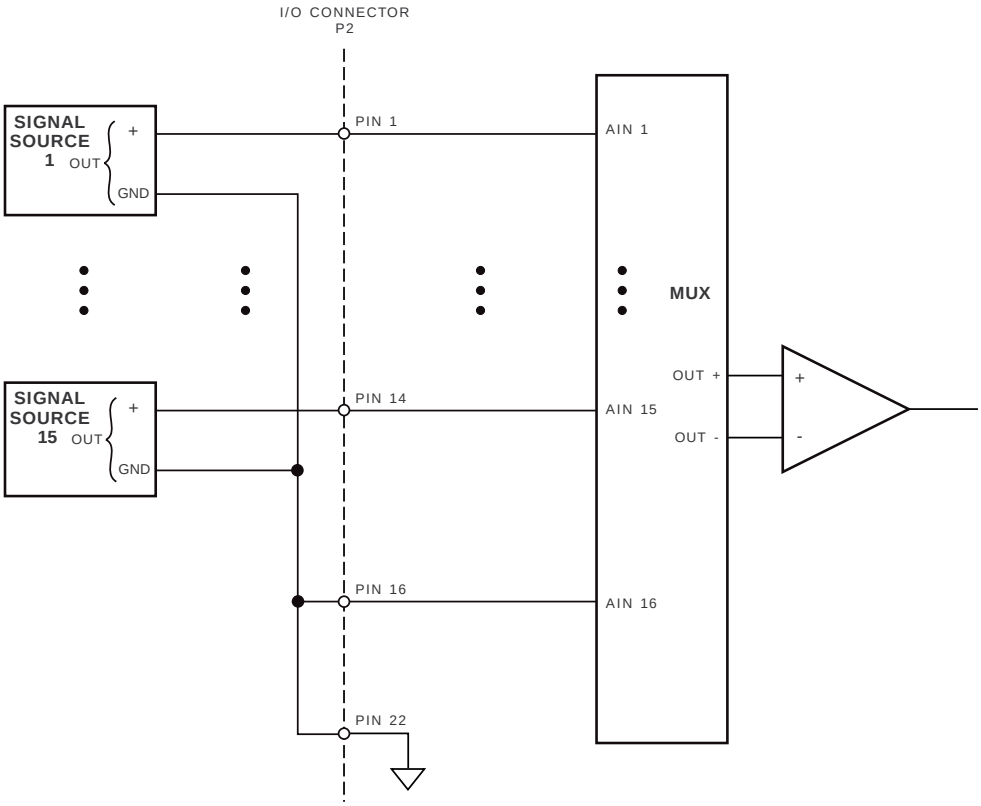


Fig. 2-2 — Analog Input Connections

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the features of the 210/5210 hardware. The major circuits are the A/D, the 8254 timer/counters, and the programmable peripheral interface which provides the digital I/O lines. Module interrupts are also described in this chapter.

The 210/5210 has three major circuits, the A/D, the timer/counters, and the 8255 programmable peripheral interface (PPI) which provides the digital I/O lines. Figure 3-1 shows the block diagram of the module. This chapter describes hardware which makes up the major circuits. It also discusses interrupts.

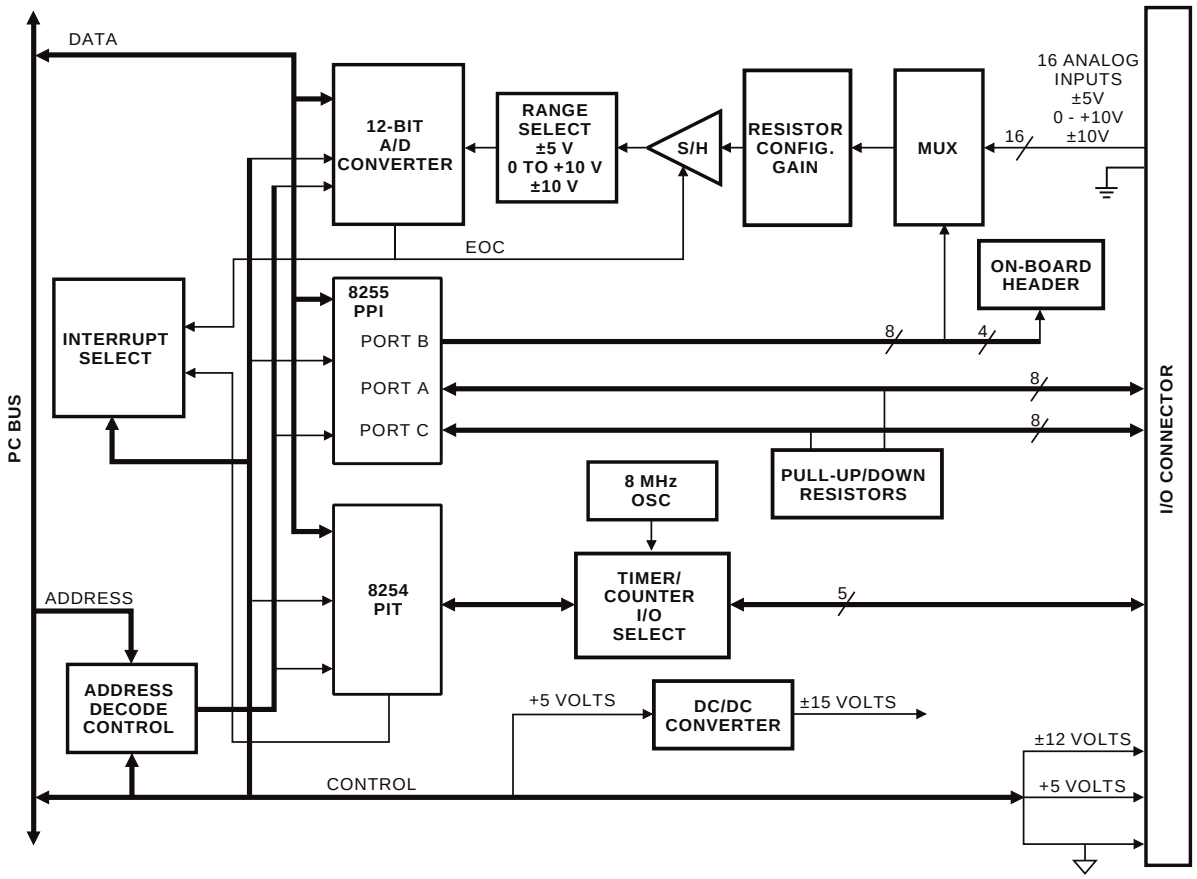


Fig. 3-1 — DM210/DM5210 Block Diagram

A/D Conversion Circuitry

The 210/5210 performs analog-to-digital conversions on up to 16 analog input channels. The following paragraphs describe the A/D circuitry.

Analog Inputs

Sixteen single-ended analog input channels are available on the 210/5210. The analog input range is jumper-selectable for -5 to +5 volts, -10 to +10 volts, or 0 to +10 volts, with ±35 Vdc overvoltage protection. The channels are connected to a sample-and-hold amplifier through a multiplexing circuit. The active channel is selected through software, as described in Chapter 4.

The S/H amplifier captures and holds the input signal at a constant level while the conversion is performed, ensuring that dynamic analog signals are accurately digitized. This capacitive circuit quickly charges to a level corresponding to the input voltage being sampled and holds the charge for the duration of the conversion.

A/D Converter

The 12-bit A/D converter , when combined with the typical acquisition time of the sample-and-hold circuitry, provides a throughput rate of up to 40,000 samples per second. The A/D output is a 12-bit data word. Note that 8-bit conversions can be performed when speed is more critical than resolution. Eight-bit conversions can increase the throughput rate to about 45 kHz.

Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8 MHz timer/counters to support a wide range of timing and counting functions. These timer/counters can be cascaded or used individually for many applications.

Each timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. The clock sources for the timer/counters can be selected using jumpers on header connector P4 (see Chapter 1). The timer/counters can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in Chapter 4. The command word also lets you set up the mode of operation. The six programmable modes are:

Mode 0 Event Counter (Interrupt on Terminal Count)
Mode 1 Hardware-Retriggerable One-Shot
Mode 2 Rate Generator
Mode 3 Square Wave Mode
Mode 4 Software-Triggered Strobe
Mode 5 Hardware Triggered Strobe (Retriggerable)

These modes are detailed in the 8254 Data Sheet, reprinted from Intel in Appendix C.

Digital I/O, Programmable Peripheral Interface

The 8255 programmable peripheral interface (PPI) is used for digital I/O functions. This high-performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each:

Group A — Port A (8 lines) and Port C Upper (4 lines);
Group B — Port B (8 lines) and Port C Lower (4 lines).

Sixteen lines, Port A, Port C Lower, and Port C Upper, are brought out to the I/O connector. Four of Port B's lines are used to control on-board functions. The remaining four Port B lines, PB4-PB7, are available at the pads labeled P8 on the module. You can use these ports in one of these three PPI operating modes:

Mode 0 — Basic input/output. Lets you use simple input and output operation for a port. Data is written to or read from the specified port.

Mode 1 — Strobed input/output. Lets you transfer I/O data from Port A in conjunction with strobes or handshaking signals.

Mode 2 — Strobed bidirectional input/output. Lets you communicate bidirectionally with an external device through Port A. Handshaking is similar to Mode 1.

These modes are detailed in the 8255 Data Sheet, reprinted from Intel in Appendix C.

Interrupts

The 210/5210 has three jumper-selectable interrupt sources: end-of-convert, 8254 timer/counter output 2, and the external clock for timer/counter 2 brought onto the module through P2. The end-of-convert signal can be used to interrupt the computer when an A/D conversion is completed. The 8254 timer/counter output 2 can be used to generate an end-of-count interrupt. The external clock 2 interrupt can be used to generate interrupts at any desired interval. Chapter 4 provides some programming information about interrupts.

CHAPTER 4

MODULE OPERATION AND PROGRAMMING

This chapter shows you how to program and use your 210/5210. It provides a complete description of the I/O map, a detailed description of programming operations and operating modes, and flow diagrams to aid you in programming. The example programs included on the disk in your module package are listed at the end of this chapter. These programs, written in Turbo C and BASIC, include source code to simplify your applications programming.

Defining the I/O Map

The I/O map for the 210/5210 is shown in Table 4-1 below. As shown, the module occupies 12 consecutive I/O port locations. The base address (designated as BA) can be selected using DIP switch S1 as described in Chapter 1, *Module Settings*. This switch can be accessed without removing the module from the connector. S1 is factory set at 300 hex (768 decimal). The following sections describe the register contents of each address used in the I/O map.

Table 4-1: DM210/DM5210 I/O Map

Register Description	Read Function	Write Function	Address * (Decimal)
8255 PPI Port A	Read Port A digital input lines	Program Port A digital output lines	BA + 0
8255 PPI Port B (Channel Select)	Read Port B bits	Program channel number; PB4-7 available for digital I/O operations	BA + 1
8255 PPI Port C	Read Port C digital input lines	Program Port C digital output lines	BA + 2
8255 PPI Control Word	Reserved	Program PPI configuration	BA + 3
8254 Timer/Counter 0	Read count value	Load count register	BA + 4
8254 Timer/Counter 1	Read count value	Load count register	BA + 5
8254 Timer/Counter 2	Read count value	Load count register	BA + 6
8254 Timer/Counter Control Word	Reserved	Program counter mode	BA + 7
Read Data/ Start 12-bit Conversion	Read A/D converted data, MSB	Start 12-bit A/D conversion	BA + 8
Read Data/ Start 8-bit Conversion	Read A/D converted data, LSB	Start 8-bit A/D conversion	BA + 9
Read Status/Clear IRQ	Read status word	Clear interrupt line	BA + 10
IRQ Enable	Reserved	Enable and disable interrupt generation	BA + 11
* BA = Base Address			

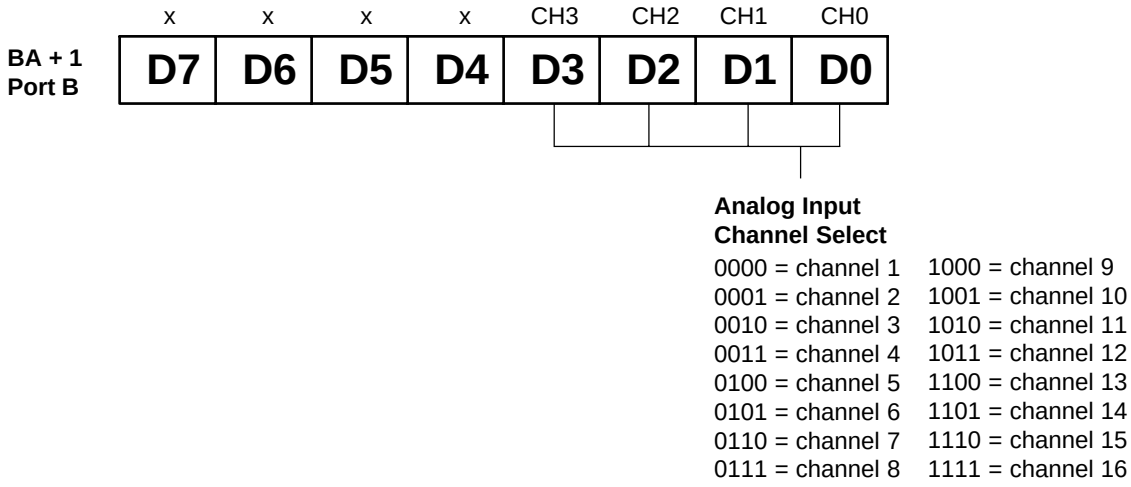
BA + 0: PPI Port A — Digital I/O (Read/Write)

Transfers the 8-bit Port A digital input and digital output data between the module and an external device. A read transfers data from the external device, through P2, and into PPI Port A; a write transfers the written data from Port A through P2 to an external device.

BA + 1: PPI Port B — Channel Select (Read/Write)

The bottom four bits, PB0-PB3, program the analog input channel. The remaining four bits, PB4-PB7, are brought out onto on-board pads, labeled P8, so that they can be used for digital control functions. Remember that if you are using these four lines for control operations, you must preserve their settings when you write to this port to change channels or enable interrupts.

Reading this register shows you the current settings.

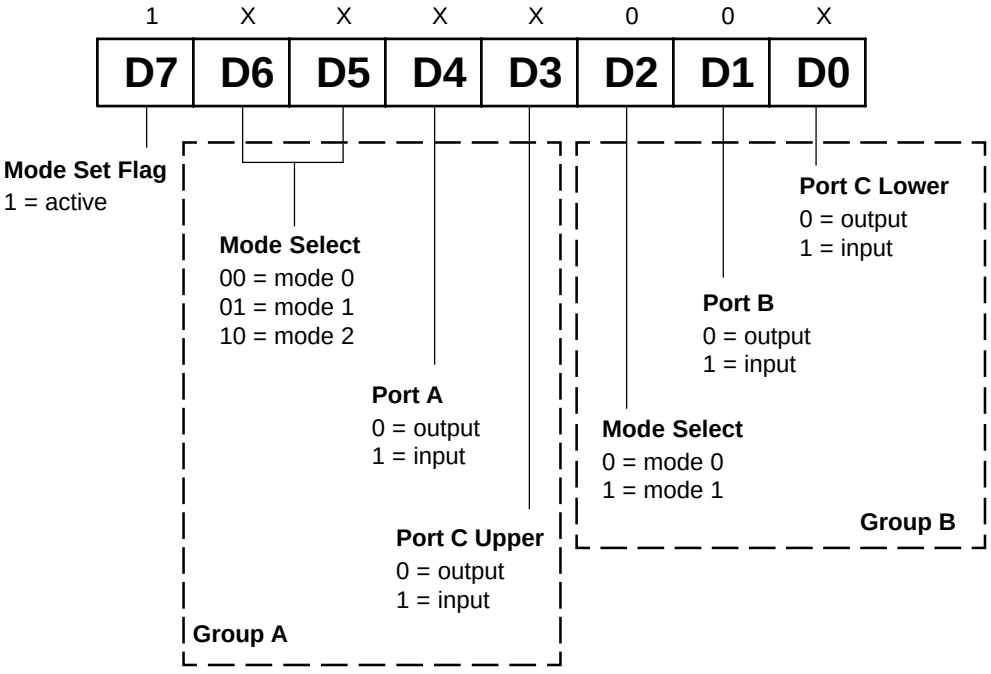

x x x x CH3 CH2 CH1 CH0
BA + 1
Port B
D7 D6 D5 D4 D3 D2 D1 D0
Analog Input
Channel Select
0000 = channel 1
0001 = channel 2
0010 = channel 3
0011 = channel 4
0100 = channel 5
0101 = channel 6
0110 = channel 7
0111 = channel 8
1000 = channel 9
1001 = channel 10
1010 = channel 11
1011 = channel 12
1100 = channel 13
1101 = channel 14
1110 = channel 15
1111 = channel 16

BA + 2: PPI Port C — Digital I/O (Read/Write)

Transfers the two 4-bit Port C digital input and digital output data groups (Port C Upper and Port C Lower) between the module and an external device. A read transfers data from the external device, through P2, and into PPI Port C; a write transfers the written data from Port C through P2 to an external device.

BA + 3: 8255 PPI Control Word (Write Only)

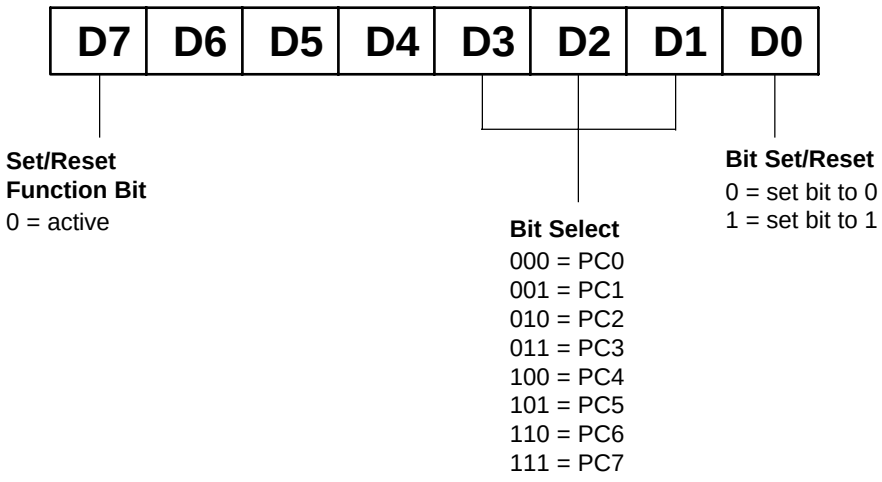
When bit 7 of this word is set to 1, a write programs the PPI configuration. The PPI must be programmed so that Port B is a Mode 0 output port, as shown below (X = don't care).



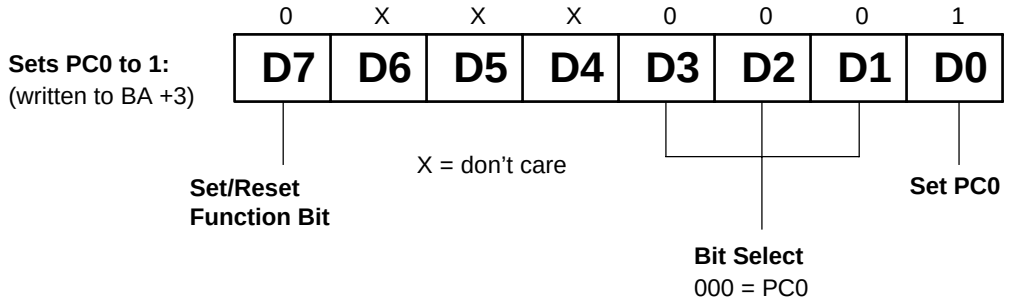
The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.

8255 Port I/O Flow Direction and Control Words, Mode 0						
Group A		Group B		Control Word		
Port A	Port C Upper	Port B	Port C Lower	Binary	Decimal	Hex
Output	Output	Output	Output	1 0 0 0 0 0 0 0	128	80
Output	Output	Output	Input	1 0 0 0 0 0 0 1	129	81
Output	Output	Input	Output	1 0 0 0 0 0 1 0	130	82
Output	Output	Input	Input	1 0 0 0 0 0 1 1	131	83
Output	Input	Output	Output	1 0 0 0 1 0 0 0	136	88
Output	Input	Output	Input	1 0 0 0 1 0 0 1	137	89
Output	Input	Input	Output	1 0 0 0 1 0 1 0	138	8A
Output	Input	Input	Input	1 0 0 0 1 0 1 1	139	8B
Input	Output	Output	Output	1 0 0 1 0 0 0 0	144	90
Input	Output	Output	Input	1 0 0 1 0 0 0 1	145	91
Input	Output	Input	Output	1 0 0 1 0 0 1 0	146	92
Input	Output	Input	Input	1 0 0 1 0 0 1 1	147	93
Input	Input	Output	Output	1 0 0 1 1 0 0 0	152	98
Input	Input	Output	Input	1 0 0 1 1 0 0 1	153	99
Input	Input	Input	Output	1 0 0 1 1 0 1 0	154	9A
Input	Input	Input	Input	1 0 0 1 1 0 1 1	155	9B

When bit 7 of the PPI control word is set to 0, a write can be used to individually program the Port C lines.



For example, if you want to set Port C bit 0 to 1, you would set up the control word so that bit 7 is 0; bits 1, 2, and 3 are 0 (this selects PC0); and bit 0 is 1 (this sets PC0 to 1). The control word is set up like this:



BA + 4: 8254 Timer/Counter 0 (Read/Write)

A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 5: 8254 Timer/Counter 1 (Read/Write)

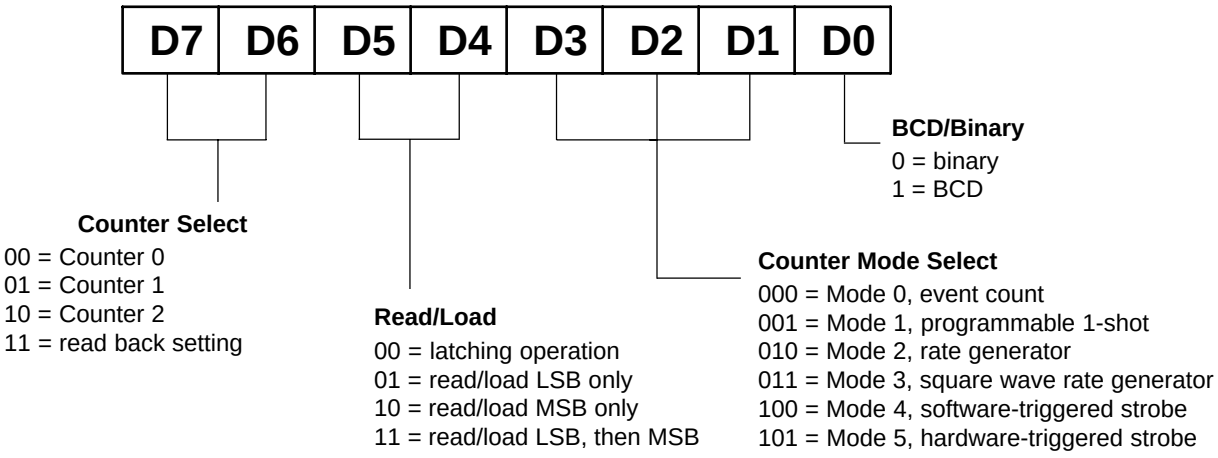
A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 6: 8254 Timer/Counter 2 (Read/Write)

A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 7: 8254 Control Word (Write Only)

Accesses the 8254 control register to directly control the three timer/counters.


D7 D6 D5 D4 D3 D2 D1 D0
BCD/Binary
0 = binary
1 = BCD
Counter Select
00 = Counter 0
01 = Counter 1
10 = Counter 2
11 = read back setting
Read/Load
00 = latching operation
01 = read/load LSB only
10 = read/load MSB only
11 = read/load LSB, then MSB
Counter Mode Select
000 = Mode 0, event count
001 = Mode 1, programmable 1-shot
010 = Mode 2, rate generator
011 = Mode 3, square wave rate generator
100 = Mode 4, software-triggered strobe
101 = Mode 5, hardware-triggered strobe

BA + 8: Read MSB Data/Start 12-Bit Conversion (Read/Write)

A read provides the MSB (8 most significant bits) of the A/D conversion, as defined below. The converted data is left-justified. When you are performing 8-bit conversions, only the MSB must be read.

Writing to this address starts a 12-bit A/D conversion (the data written is irrelevant).

MSB	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
12-Bit:	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4
8-Bit:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

BA + 9: Read LSB Data/Start 8-Bit Conversion (Read/Write)

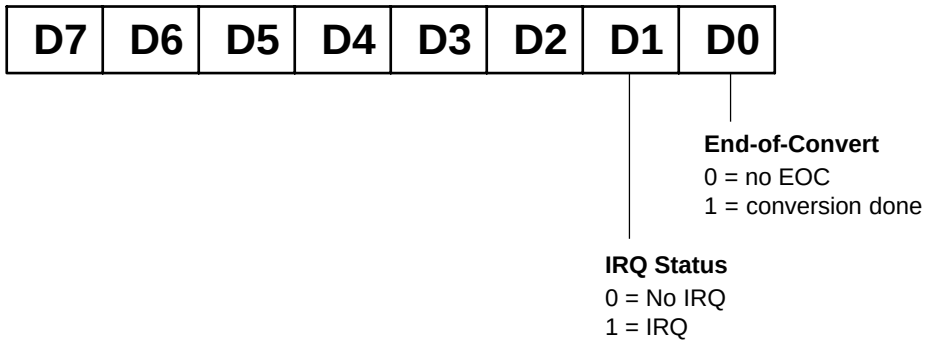
A read provides the LSB (4 least significant bits) of the A/D conversion, as defined below. The converted data is left-justified.

Writing to this address starts an 8-bit A/D conversion (the data written is irrelevant).

LSB	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
	Bit 3	Bit 2	Bit 1	Bit 0	X	X	X	X

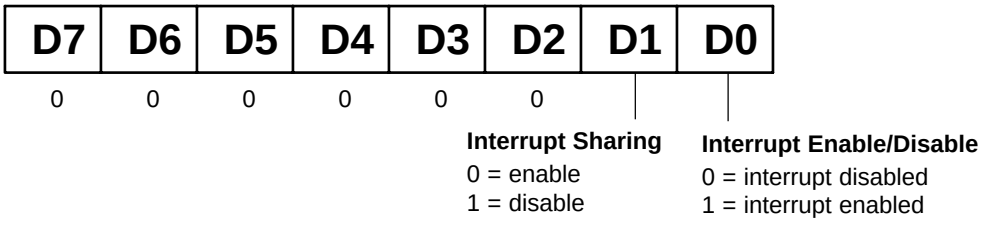
BA + 10: Read Status/Clear IRQ (Read/Write)

A read provides the two status bits defined below. The end-of-convert bit goes high when a conversion is complete. The IRQ status bit goes high when an interrupt has occurred and stays high until a clear IRQ command is sent. The clear IRQ command is sent by writing to BA + 10 (data written is irrelevant).



BA + 11: IRQ Enable (Write Only)

A write enables and disables interrupt generation and interrupt sharing. Writing a “1” to bit 0 enables interrupt generation; writing a “0” disables interrupt generation, writing a “1” to bit 1 disables interrupt sharing;writing a “0” enables interrupt sharing. At power up this register is set to "0".



Programming the DM210/DM5210

This section gives you some general information about programming and the 210/5210, and then walks you through the major 210/5210 programming functions. These descriptions will help you as you use the example programs included with the module and the programming flow diagrams at the end of this chapter. All of the program descriptions in this section use decimal values unless otherwise specified.

The 210/5210 is programmed by writing to and reading from the correct I/O port locations on the module. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

Language	Read	Write
BASIC	Data=INP(Address)	OUT Address,Data
Turbo C	Data=inportb(Address)	outportb(Address,Data)
Turbo Pascal	Data:=Port[Address]	Port[Address]:=Data

Assembly	mov dx,Address
	in al,dx
		mov dx,Address
		mov al,Data
		out dx,al

In addition to being able to read/write the I/O ports on the 210/5210, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal, C, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

Language	Modules	Integer Division	AND	OR
C	% a = b % c	/ a = b / c	& a = b & c	| a = b | c
Pascal	MOD a := b MOD c	DIV a := b DIV c	AND a := b AND c	OR a := b OR c
BASIC	MOD a = b MOD c	\ a = b \ c	AND a = b AND c	OR a = b OR c

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with the 210/5210!**

Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

Example: Clear bit 5 in a port. Read in the current value of the port, AND it with 223 ($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V = INP(PortAddress)
V = V AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

Example: Set bit 3 in a port. Read in the current value of the port, OR it with 8 ($8 = 2^3$), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V := Port[PortAddress];
V := V OR 8;
Port[PortAddress] := V;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 - (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

Example: Clear bits 2, 4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 171;
outportb(port_address, v);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

Example: Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov dx, PortAddress
in al, dx
or al, 168
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

Example: Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 199;
v = v | 40;
outportb(port_address, v);
```

A final note: Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 (2^5) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 *is* already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

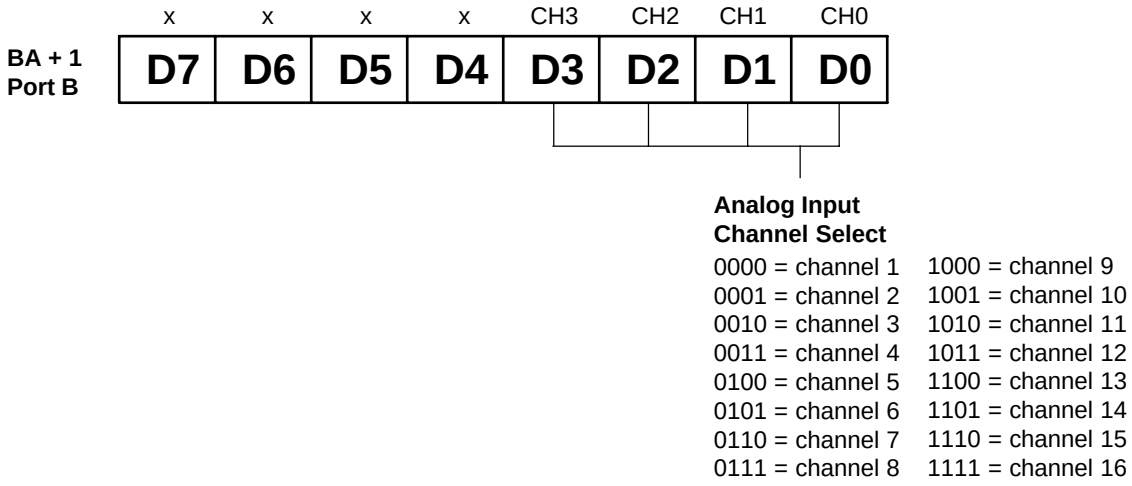
Now that you know how to clear and set bits, we are ready to look at the programming steps for the 210/5210 module functions.

A/D Conversions

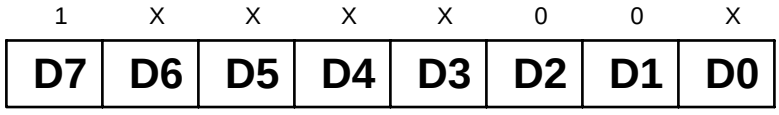
The following paragraphs walk you through the programming steps for performing A/D conversions. Detailed information about the conversion modes is presented in this section. You can follow these steps on the flow diagrams at the end of this chapter and in our example programs included with the module. In this discussion, BA refers to the base address.

• Initializing the 8255 PPI

Four of the eight 8255 Port B lines are used to control the channel selection for taking a reading. Port B is programmed at I/O address location BA + 1:


x x x x CH3 CH2 CH1 CH0
BA + 1
Port B
D7 D6 D5 D4 D3 D2 D1 D0
Analog Input
Channel Select
0000 = channel 1
0001 = channel 2
0010 = channel 3
0011 = channel 4
0100 = channel 5
0101 = channel 6
0110 = channel 7
0111 = channel 8
1000 = channel 9
1001 = channel 10
1010 = channel 11
1011 = channel 12
1100 = channel 13
1101 = channel 14
1110 = channel 15
1111 = channel 16

To use Port B for these control functions, the 8255 must be initialized so that Port B is set up as a Mode 0 output port. This is done by writing this data to the PPI control word at I/O address BA + 3 (X = don't care):



The top four bits of Port B are brought out to on-board pads where they are available for your use. Keep in mind that when you are programming some of the bits in this port, you may need to preserve the state of other bits.

• Selecting a Channel

To select a conversion channel, you must assign values to bits 0 through 3 in the PPI Port B port at BA + 1. The bit structure diagram above shows you the four-bit instruction for each of the 16 channels.

• Enabling and Disabling Interrupts

Any time you use interrupts, this bit, bit 1 at port BA + 11, must be set high to enable the IRQ circuitry.

• Starting an A/D Conversion

A/D conversions are started by writing to the appropriate I/O port. For 12-bit conversions, Port BA + 8 is used. For 8-bit conversions, Port BA + 9 is used. A START CONVERT command must be issued for each A/D conversion. The data written to start a conversion is irrelevant. Figure 4-1 shows the timing diagram for A/D conversions.

• Channel Scanning

If you want to sample a sequence of channels, you can set up the 210/5210 for channel scanning. The main concern when you scan channels is that you allow enough settling time between the selection of the channel and the start of the A/D conversion. The channel scanning flow diagram at the end of this chapter explains how to properly program for channel scanning and avoid settling time problems.

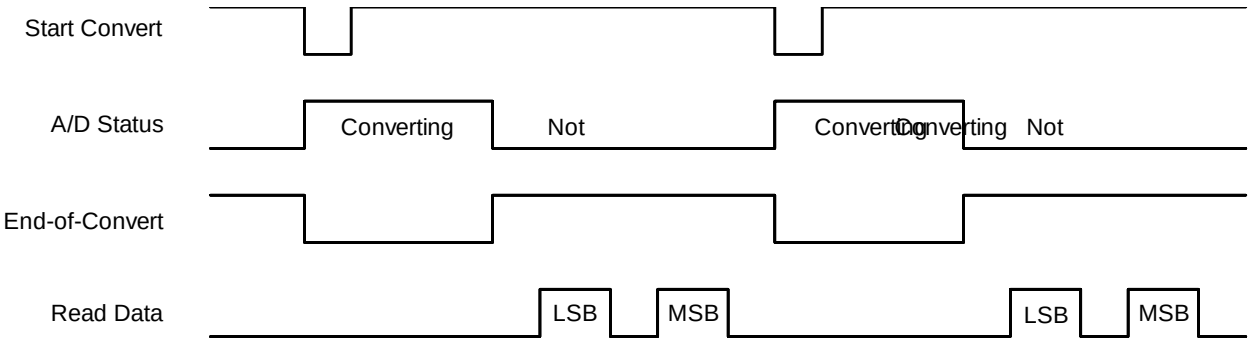


Fig. 4-1 — A/D Conversion Timing Diagram

• Monitoring Conversion Status

The A/D conversion status can be monitored through the end-of-convert (EOC) signal. This signal, the inverse of the STATUS signal output by the A/D converter, is low when a conversion is in progress and goes high when the conversion is completed. This low-to-high transition can be used to generate an interrupt.

• Reading the Converted Data

The general algorithm for taking an A/D reading is:

1. Start a 12-bit conversion by writing to BA + 8:

```
out base_address+8,0
```

(Note that the value you send is not important. The act of writing to this I/O location is the key to starting a conversion.)

2. Delay at least 20 microseconds or monitor end-of-convert for a transition, or use an interrupt.
3. Read the least significant byte of the converted data from BA + 9:

```
lsb% = inp(base_address% +9)
```

4. Read the most significant byte of the converted data from BA + 8:

```
msb% = inp(base_address% +8)
```

5. Combine them into the 12-bit result by shifting the LSB four bits to the right. The MSB must also be weighted correctly:

```
result% = (msb% * 16) + (lsb%/16)
```

For a 12-bit conversion, the A/D data read is left justified in a 16-bit word, with the least significant four bits equal to zero. Because of this, the two bytes of A/D data read must be scaled to obtain a valid A/D reading. For example, for a voltage range of ±5 volts, once the reading is calculated, it can be correlated to a voltage value by subtracting 2048 to scale it and then multiplying by 2.4414 millivolts.

For example, if the A/D reading is 1024, the analog input voltage is calculated as follows:

```
(1024 - 2048) bits * 2.4414 mV/bit = -2.49999 volts.
```

Note that 8-bit A/D conversions can also be performed by writing to I/O location BA + 9 to start a conversion. While an 8-bit conversion has a lower resolution, it is performed more rapidly, since the converted data is contained in a single byte.

The key digital codes and their input voltage values are given for 12-bit and 8-bit conversions in the following two tables.

12-Bit A/D Code Table			
Input Voltage Range			
0 to +10 Volts	-10 to +10 Volts	-5 to +5 Volts	Output Code
+9.9976 volts	+9.9951 volts	+4.9976 volts	MSB 1111 1111 1111 LSB
+7.500 volts	+5.000 volts	+2.500 volts	1100 0000 0000
+5.000 volts	0 volts	0 volts	1000 0000 0000
+2.500 volts	-5.000 volts	-2.500 volts	0100 0000 0000
0 volts	-10.000 volts	-5.000 volts	0000 0000 0000
For 0 to +10 & ±5 volts, 1 LSB = 2.44 millivolts; for ±10 volts, 1 LSB = 4.88 millivolts.			

8-Bit A/D Code Table			
Input Voltage Range			
0 to +10 Volts	-10 to +10 Volts	-5 to +5 Volts	Output Code
+9.9609 volts	+9.9219 volts	+4.9609 volts	MSB 1111 1111 LSB
+7.500 volts	+5.000 volts	+2.500 volts	1100 0000
+5.000 volts	0 volts	0 volts	1000 0000
+2.500 volts	-5.000 volts	-2.500 volts	0100 0000
0 volts	-10.000 volts	-5.000 volts	0000 0000
For 0 to +10 & ±5 volts, 1 LSB = 39.063 millivolts; for ±10 volts, 1 LSB = 78.126 millivolts.			

Interrupts

• What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard 'interrupts' the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your 210/5210 can interrupt the processor when a variety of conditions are met. By using these interrupts, you can write software that effectively deals with real world events.

• Interrupt Request Lines

To allow different peripheral devices to generate interrupts on the same computer, the PC bus has eight different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by the PC's interrupt controller. The interrupt controller checks to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted

if the second request has a higher priority. The priority level is based on the number of the IRQ; IRQ0 has the highest priority, IRQ1 is second-highest, and so on through IRQ7, which has the lowest. Many of the IRQs are used by the standard system resources. IRQ0 is used by the system timer, IRQ1 is used by the keyboard, IRQ3 by COM2, IRQ4 by COM1, and IRQ6 by the disk drives. Therefore, it is important for you to know which IRQ lines are available in your system for use by the 210/5210.

• 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in the PC is the 8259 Programmable Interrupt Controller. To use interrupts, you need to know how to read and set the 8259's interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to the 8259.

• Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. If a bit is **set** (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is **clear** (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR is programmed through port 21H.

IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

I/O Port 21H

For all bits:
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

• End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 interrupt controller must be notified. This is done by writing the value 20H to I/O port 20H.

• What Exactly Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the 210/5210), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

• Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the INTRPTS source code included on your 210/5210 program disk for a better understanding of interrupt program development.

• Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the

stack **BEFORE** you do anything else. Second, just before exiting your ISR, you must clear the interrupt status of the 210/5210 and write an end-of-interrupt command to the 8259 controller. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by interrupt programming, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers **do not** automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

NOTE: If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.

There are a few cautions you must consider when writing your ISR. The most important is, **do not use any DOS functions or routines that call DOS functions from within an ISR**. DOS is **not** reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Clear the interrupt bit on the 210/5210 by writing any value to BA + 10.
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H.
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like:

In C:

```
void interrupt ISR(void)
{
   /* Your code goes here. Do not use any DOS functions! */
   outportb(BaseAddress + 10, 0);   /* Clear 210/5210 interrupt */
   outportb(0x20, 0x20);            /* Send EOI command to 8259 */
}
```

In Pascal:

```
Procedure ISR; Interrupt;
begin
   { Your code goes here. Do not use any DOS functions! }
   Port[BaseAddress + 10] := 0;     { Clear 210/5210 interrupt }
   Port[$20] := $20;                { Send EOI command to 8259 }
end;
```

• Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the 210/5210 will be using IRQ3, you should save the value of interrupt vector 11.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21H and **set** the bit that corresponds to your IRQ (remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21H.

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vector 8 is for IRQ0, vector 9 is for IRQ1, and so on.

If you need to program the source of your interrupts, do that next. For example, if you are using a programmable interval timer to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

• Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in when your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H. Restore the interrupt vector that was saved at startup with either DOS function 35H (get interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

• Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15, even though the corresponding IRQs are numbered 0 through 7.
- Two of the most common mistakes when writing an ISR are forgetting to clear the interrupt status of the 210/5210 and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8-MHz timer/counters for timing and counting functions such as frequency measurement, event counting, and interrupts. All three timer/counters are cascaded at the factory. Figure 4-2 shows the timer/counter circuitry.

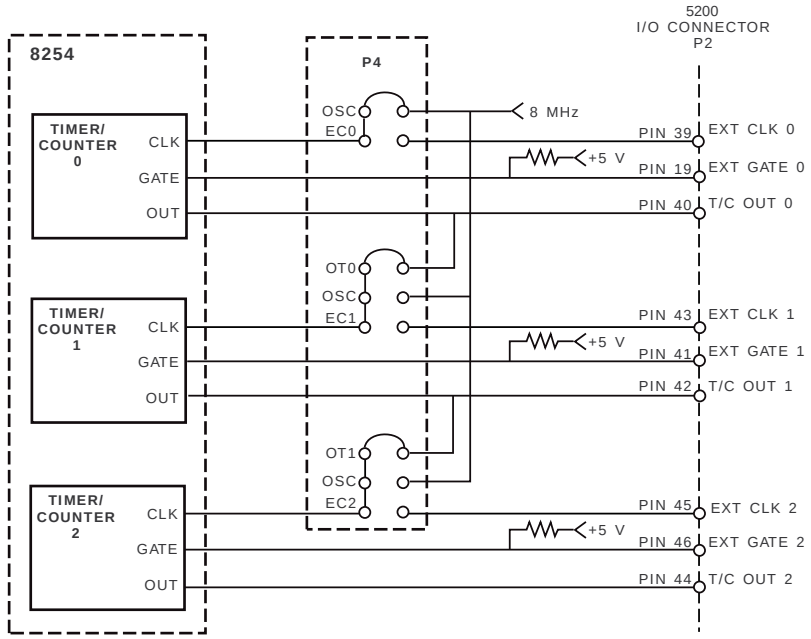


Fig. 4-2 — 8254 Programmable Interval Timer Circuit Block Diagram

Each timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. They can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in the I/O map section at the beginning of this chapter.

One of two clock sources, the on-board 8-MHz crystal or an external clock can be selected as the clock input to each timer/counter. In addition, the timer/counters can be cascaded by connecting TC0's output to TC1's clock input and TC1's output to TC2's clock input. The diagram shows how these clock sources are connected to the timer/counters.

An external gate source can be connected to each timer/counter through the I/O connector. When a gate is disconnected, an on-board pull-up resistor automatically pulls the gate high, enabling the timer/counter.

The output from each timer/counter is available at the I/O connector, where it can be used for interrupt generation or for counting functions.

The timer/counters can be programmed to operate in one of six modes, depending on your application. The following paragraphs briefly describe each mode.

Mode 0, Event Counter (Interrupt on Terminal Count). This mode is typically used for event counting. While the timer/counter counts down, the output is low, and when the count is complete, it goes high. The output stays high until a new Mode 0 control word is written to the timer/counter.

Mode 1, Hardware-Retriggerable One-Shot. The output is initially high and goes low on the clock pulse following a trigger to begin the one-shot pulse. The output remains low until the count reaches 0, and then goes high and remains high until the clock pulse after the next trigger.

Mode 2, Rate Generator. This mode functions like a divide-by-N counter and is typically used to generate a real-time clock interrupt. The output is initially high, and when the count decrements to 1, the output goes low for one clock pulse. The output then goes high again, the timer/counter reloads the initial count, and the process is repeated. This sequence continues indefinitely.

Mode 3, Square Wave Mode. Similar to Mode 2 except for the duty cycle output, this mode is typically used for baud rate generation. The output is initially high, and when the count decrements to one-half its initial count, the output goes low for the remainder of the count. The timer/counter reloads and the output goes high again. This process repeats indefinitely.

Mode 4, Software-Triggered Strobe. The output is initially high. When the initial count expires, the output goes low for one clock pulse and then goes high again. Counting is "triggered" by writing the initial count.

Mode 5, Hardware Triggered Strobe (Retriggerable). The output is initially high. Counting is triggered by the rising edge of the gate input. When the initial count has expired, the output goes low for one clock pulse and then goes high again.

Digital I/O

The 20 available 8255 PPI-based digital I/O lines can be used to transfer data between the computer and external devices. The digital input lines of Ports A and C can have pull-up or pull-down resistors installed, as described in Chapter 1.

Example Programs and Flow Diagrams

Included with the 210/5210 is a set of example programs that demonstrate the use of many of the module's features. These examples are in written in C and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 5210DIAG, which is especially helpful when you are first checking out your module after installation and when calibrating the module (Chapter 5).

Before using the software included with your module, make a backup copy of the disk. You may make as many backups as you need.

C Programs

These programs are source code files so that you can easily develop your own custom software for the 210/5210. In the C directory, DM5210.H and DM5210.INC contain all the functions needed to implement the main C programs. H defines the addresses and INC contains the routines called by the main programs.

Analog-to-Digital:

SOFTTRIG	Demonstrates how to use a software trigger for acquiring data.

Timer/Counters:

TIMER	A short program demonstrating how to program the 8254 for use as a timer.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

Interrupts:

INTRPTS	Shows the bare essentials required for using interrupts.
INTSTR	A complete program showing interrupt-based streaming to disk.

BASIC Programs

These programs are source code files so that you can easily develop your own custom software for the 210/5210.

Analog-to-Digital:

SINGLE	Demonstrates how to perform single conversions.
SCAN	Demonstrates how to change channels while acquiring data.

Timer/Counters:

TIMER	A short program demonstrating how to program the 8254 for use as a timer.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

Flow Diagrams

The following paragraphs provide a description and flow diagrams for performing A/D conversions and channel scanning. These diagrams will help you to build your own custom applications programs.

• Single Convert Flow Diagram (Figure 4-3)

This flow diagram shows you the steps for taking a single sample on a selected channel. A sample is taken each time you send the Start Convert command. All of the samples will be taken on the same channel and until you change the channel by writing a new value to the bottom four bits in the PPI Port B register (BA + 1). Changing this value before each Start Convert command is issued lets you take the next reading from a different channel.

• Channel Scanning Flow Diagram (Figure 4-4)

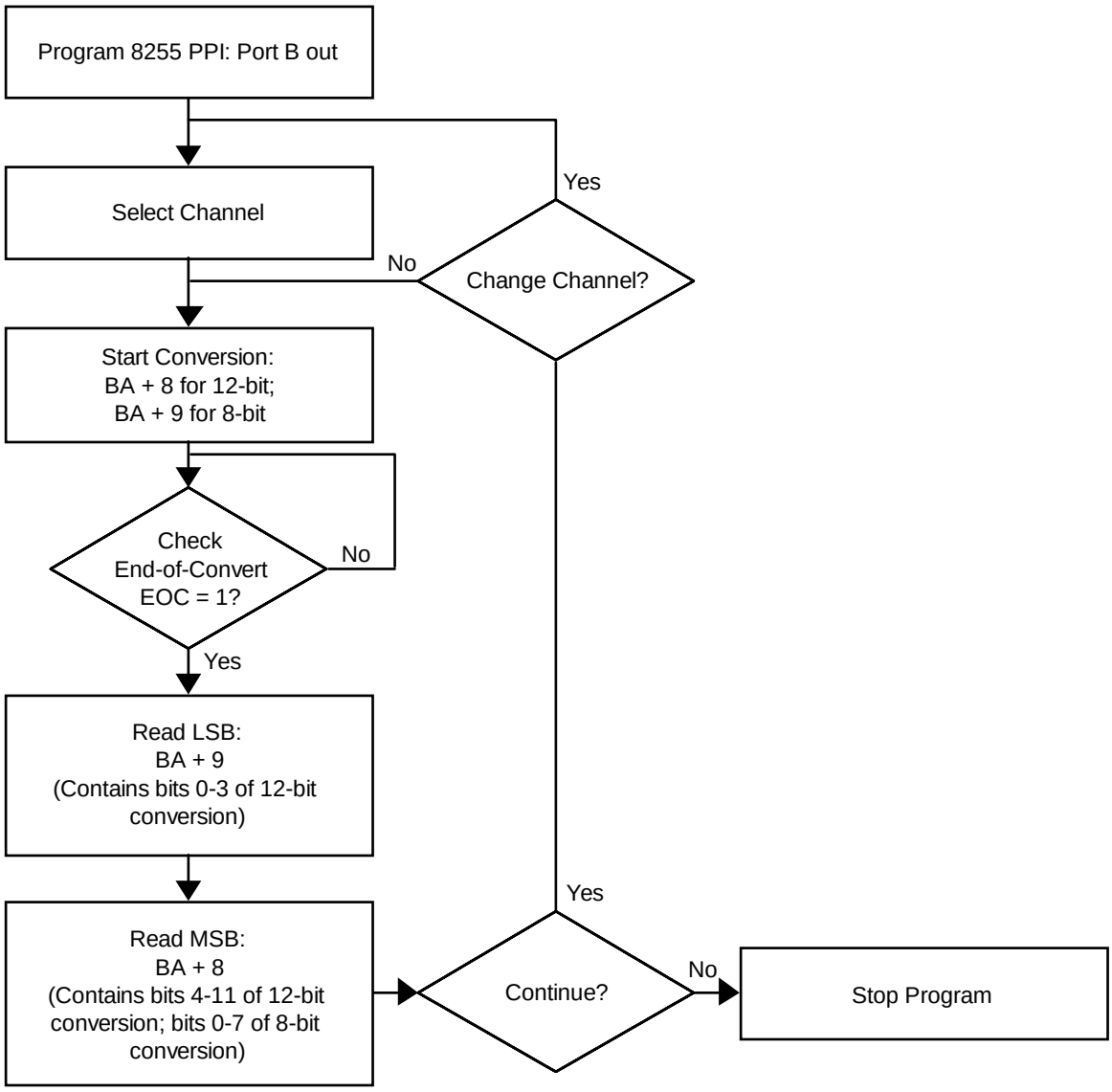


Fig. 4-3 — Single Conversion Flow Diagram

This flow diagram shows you the steps for taking a single sample on a sequence of channels. After programming Port B as a Mode 0 output port, you select the starting channel in your sequence of channels to be scanned. After making your initial channel selection, you must allow for enough of a delay in your program for the selected channel to settle before starting the first A/D conversion. As soon as the first conversion is started, you can then immediately select your next channel in the sequence. Once the conversion is started, the signal on the sampled channel has been "locked in", and you do not have to wait for an end-of-convert transition before programming the next channel. Selecting the next channel as soon as the conversion of the previous channel is started ensures that enough time is allowed for the new channel to settle before the next conversion is started, regardless of your PC type. Except for the initial delay between the starting channel selection and first conversion, you do not have to be concerned with building delays into your program and the accuracy of the conversions when following this program structure. Note that the data you read in the Read LSB, Read MSB steps will always be the data from the previously selected channel, not the data from the channel selected in the Select Next Channel block.

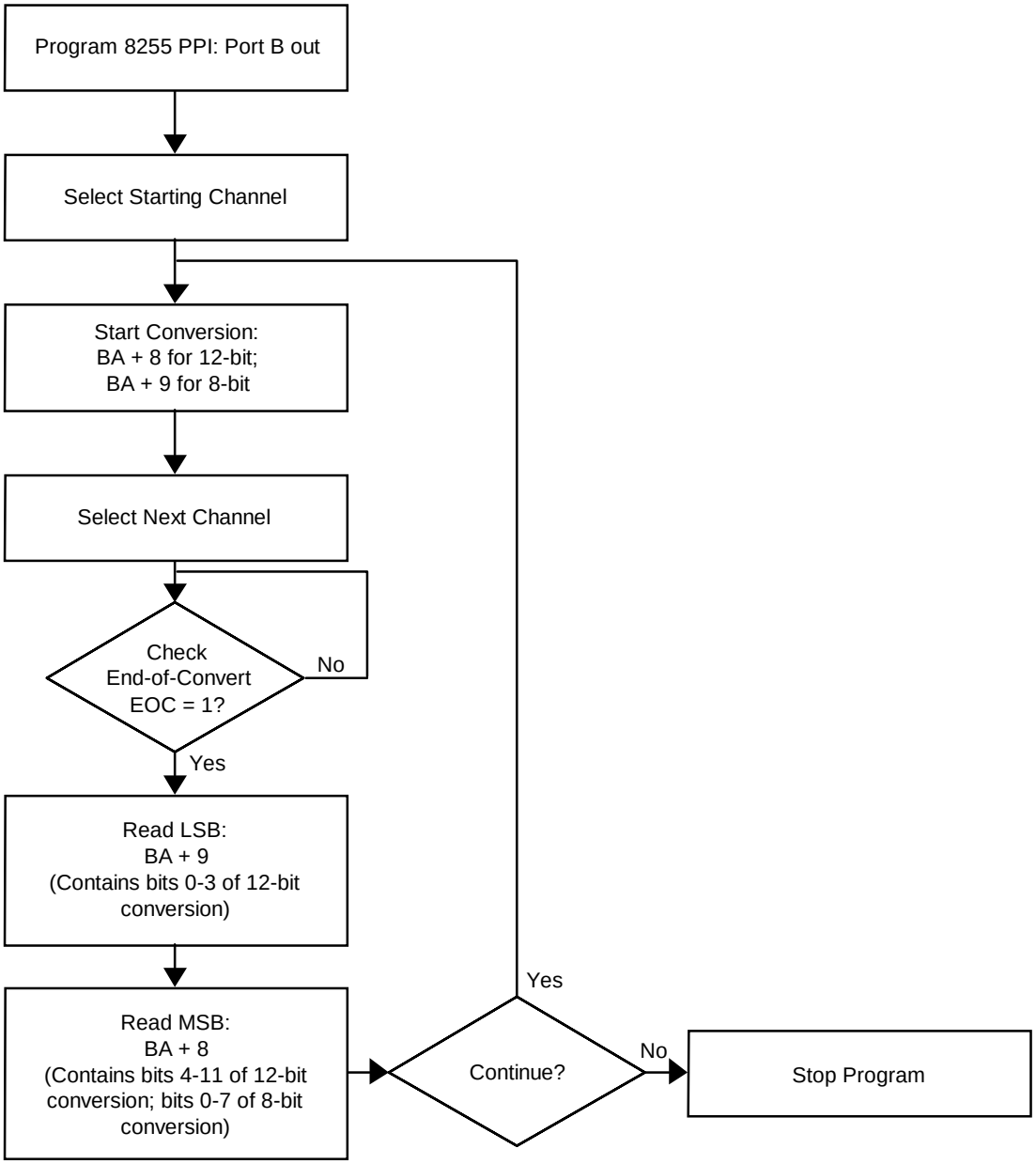


Fig. 4-4 — Channel Scanning Flow Diagram

CHAPTER 5

CALIBRATION

This chapter tells you how to calibrate the 210/5210 using the 5210DIAG calibration program included in the example software package and the three trimpots on the module. These trimpots calibrate the A/D converter gain and offset.

This chapter tells you how to calibrate the A/D converter gain and offset. The offset and full-scale performance of the module's A/D converter is factory-calibrated. Any time you suspect inaccurate readings, you can check the accuracy of your conversions using the procedure below, and make adjusts as necessary. Using the 5210DIAG diagnostics program is a convenient way to monitor conversions while you calibrate the module.

Calibration is done with the module installed in your system. You can access the trimpots at the edge of the module. Power up the system and let the board circuitry stabilize for 15 minutes before you start calibrating.

Required Equipment

The following equipment is required for calibration:

- Precision Voltage Source: -10 to +10 volts
- Digital Voltmeter: 5-1/2 digits
- Small Screwdriver (for trimpot adjustment)

While not required, the 5210DIAG diagnostics program (included with example software) is helpful when performing calibrations. Figure 5-1 shows the module layout with the three trimpots used for calibration (TR1, TR2, and TR4) located along the top right edge.

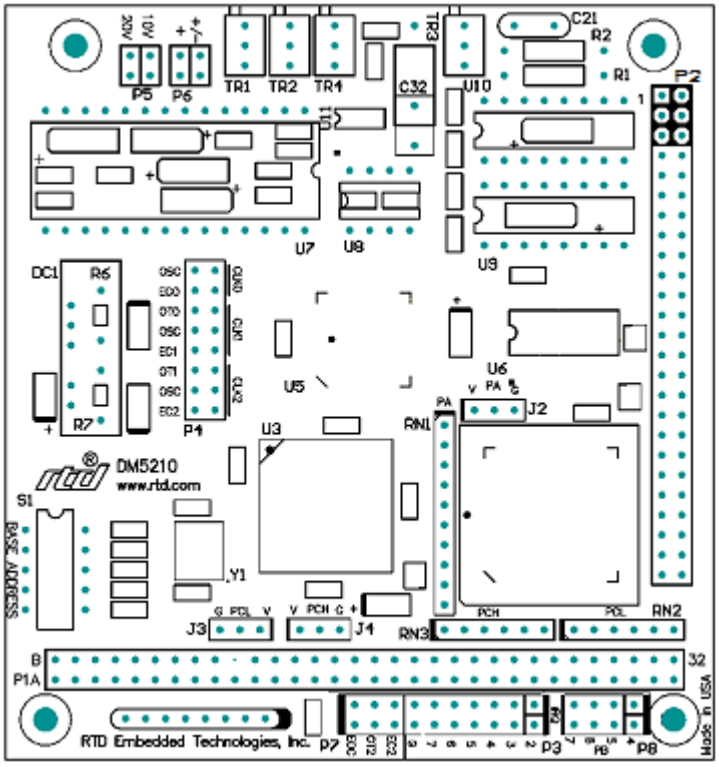


Fig. 5-1 — Module Layout

A/D Calibration

Two procedures are used to calibrate the A/D converter for all input voltage ranges. The first procedure calibrates the converter for the unipolar range (0 to +10 volts), and the second procedure calibrates the bipolar ranges (±5, ±10 volts). Table 5-1 shows the ideal input voltage for each bit weight for all three input ranges.

Unipolar Calibration

Two adjustments are made to calibrate the A/D converter for the unipolar range of 0 to +10 volts. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot TR4 is used to make the offset adjustment, and trimpot TR1 is used for gain adjustment. This calibration procedure is performed with the module set up for a 0 to +10 volt input range. Before making these adjustments, make sure that the jumpers on P5 and P6 are set for this range.

Use analog input channel 1 (gain = 1) while calibrating the module. Connect your precision voltage source to channel 1. Set the voltage source to +1.22070 millivolts, start a conversion, and read the resulting data. Adjust trimpot TR4 until it flickers between the values listed in the table at the top of the next page. Next, set the voltage to +9.49829 volts, and repeat the procedure, this time adjusting TR1 until the data flickers between the values in the table.

Table 5-1: A/D Converter Bit Weights for All Input Ranges

	Ideal Input Voltage (millivolts)		
A/D Bit Weight	**-5 to +5 Volts**	**-10 to +10 Volts**	**0 to +10 Volts**
4095 (full-scale)	+4997.56	+9995.12	+9997.56
2048	0000.00	0000.00	+5000.00
1024	-2500.00	-5000.00	+2500.00
512	-3750.00	-7500.00	+1250.00
256	-4375.00	-8750.00	+625.00
128	-4687.50	-9375.00	+312.50
64	-4843.75	-9687.50	+156.25
32	-4921.88	-9843.75	+78.13
16	-4960.94	-9921.88	+39.06
8	-4980.47	-9960.94	+19.53
4	-4990.23	-9980.47	+9.77
2	-4995.12	-9990.23	+4.88
1	-4997.56	-9995.12	+2.44
0	-5000.00	-10000.00	0.00

Data Values for Calibrating Unipolar Range (0 to +10 volts)		
	Offset (TR4) Input Voltage = +1.22070 mV	Converter Gain (TR1) Input Voltage = +9.99634 V
A/D Converted Data	0000 0000 0000 0000 0000 0001	1111 1111 1110 1111 1111 1111

Bipolar Calibration

Two adjustments are made to calibrate the A/D converter for the bipolar ranges of ±5 and ±10 volts. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot TR2 is used to make the offset adjustment, and trimpot TR1 is used for gain adjustment. These adjustments are made with the module set for a range of -5 to +5 volts. Before making these adjustments, make sure that the jumpers on P5 and P6 are set for this range.

Use analog input channel 1 (gain = 1) while calibrating the module. Connect your precision voltage source to channel 1. Set the voltage source to -4.99878 volts, start a conversion, and read the resulting data. Adjust trimpot TR2 until it flickers between the values listed in the table below. Next, set the voltage to +4.99634 volts, and repeat the procedure, this time adjusting TR1 until the data flickers between the values in the table.

Data Values for Calibrating Bipolar Ranges (Using -5 to +5 volts)		
	Offset (TR2) Input Voltage = -4.99878V	Converter Gain (TR1) Input Voltage = +4.99634V
A/D Converted Data	0000 0000 0000 0000 0000 0001	1111 1111 1110 1111 1111 1111

APPENDIX A

DM210/DM5210 SPECIFICATIONS

DM210/DM5210 Characteristics Typical @ 25° C

Interface

Switch-selectable base address, I/O mapped
Jumper-selectable interrupts

Analog Input

16 single-ended inputs
Input impedance, each channel ..... >10 megohms
Input ranges ..... ±5, ±10, 0 to +10 volts
Overvoltage protection ..... ±35 Vdc
Settling time ..... 1 µsec, max

A/D Converter AD574

Type ..... Successive approximation
Resolution ..... 12 bits (2.44 mV @ 10V; 4.88 mV @ 20V)
Linearity ..... ±1 LSB, typ
Conversion speed ..... 20 µsec, typ
Sample-and-hold acquisition time ..... 5 µsec, typ
Maximum throughput ..... 40 kHz

Digital I/O ..... CMOS 82C55 (Optional NMOS 8255)

Number of lines ..... 20 (16 at I/O connector & 4 on board)
High-level output voltage ..... 4.2V, min
Low-level output voltage ..... 0.45V, max
High-level input voltage ..... 2.2V, min; 5.5V, max
Low-level input voltage ..... -0.3V, min; 0.8V, max
High-level output current, Isource ..... 100 µA, max
Low-level output current, Isink ..... 1.7 mA, max
Darlington drive current, I(DAR) ..... -1.0 mA, min; -5.0 mA, max
(Available on any 8 pins from port B and port C)
Input load current ..... ±10 µA
Input capacitance,
C(IN)@F=1MHz ..... 10 pF
Output capacitance,
C(OUT)<@F=1MHz ..... 20 pF

Timer/Counter ..... CMOS 82C54 (Optional NMOS 8254)

Three 16-bit down counters; binary or BCD counting
Programmable operating modes (6) ..... Interrupt on terminal count; programmable one-shot; rate generator; square wave rate generator; software-triggered strobe; hardware-triggered strobe
Counter input source ..... External clock (8 MHz, max) or on-board 8-MHz clock
Counter outputs ..... Available externally; used as PC interrupts or cascaded to adjacent counter
Counter gate source ..... External gate or always enabled

Miscellaneous Inputs/Outputs (PC bus-sourced)

±5 volts, +12 volts (if supplied by computer), ground

Current Requirements

DM210: 75 mA @ +5 volts; 12 mA @ +12 volts; 28 mA @ –12 volts (.83W)
DM5210: 240 mA @ +5 volts (1.2W)

Connector

50-pin right angle header

Environmental

Operating temperature ..... 0 to +70°C
Storage temperature ..... -40 to +85°C
Humidity ..... 0 to 90% non-condensing

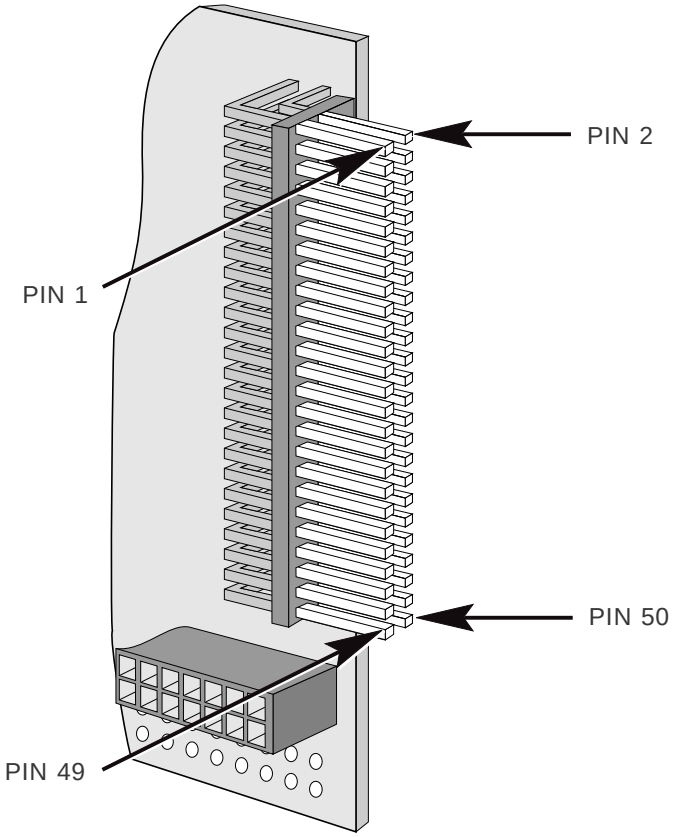
Size

3.55"L x 3.775"W x 0.6"H (90mm x 96mm x 15mm)

APPENDIX B

P2 CONNECTOR PIN ASSIGNMENTS

AIN1	1	2	AIN9
AIN2	3	4	AIN10
AIN3	5	6	AIN11
AIN4	7	8	AIN12
AIN5	9	10	AIN13
AIN6	11	12	AIN14
AIN7	13	14	AIN15
AIN8	15	16	AIN16
ANALOG GND	17	18	ANALOG GND
EXT GATE 0	19	20	ANALOG GND
ANALOG GND	21	22	ANALOG GND
PA7	23	24	PC7
PA6	25	26	PC6
PA5	27	28	PC5
PA4	29	30	PC4
PA3	31	32	PC3
PA2	33	34	PC2
PA1	35	36	PC1
PA0	37	38	PC0
EXT CLK 0	39	40	T/C OUT 0
EXT GATE 1	41	42	T/C OUT 1
EXT CLK 1	43	44	T/C OUT 2
EXT CLK 2	45	46	EXT GATE 2
+12 VOLTS	47	48	+5 VOLTS
-12 VOLTS	49	50	DIGITAL GND



P2 Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

APPENDIX C

COMPONENT DATA SHEETS

Intel 82C54 Programmable Interval Timer
Data Sheet Reprint

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

APPENDIX D WARRANTY AND RETURN POLICY

Return Policy

If you wish to return a product to the factory for service, please follow this procedure:

Read the Limited Warranty to familiarize yourself with our warranty policy.

Contact the factory for a Return Merchandise Authorization (RMA) number.

Please have the following available:

- Complete board name
- Board serial number
- A detailed description of the board's behavior

List the name of a contact person, familiar with technical details of the problem or situation, **along with their phone and fax numbers, address, and e-mail address** (if available).

List your shipping address!!

Indicate the shipping method you would like used to return the product to you.
We will not ship by next-day service without your pre-approval.

Carefully package the product, using proper anti-static packaging.

Write the RMA number in large (1") letters on the outside of the package.

Return the package to:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA

LIMITED WARRANTY

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, “acts of God” or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER’S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com

DM210/DM5210 Board User-Selected Set	
Base I/O Address:	
(hex)	(decimal)
IRQ Source & Channel Selected:	
Source (P7):	IRQ Channel # (P3):